Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
SCIENZE DELL'INGEGNERIA

CICLO XXXVII

COORDINATORE Prof. Trillo Stefano

Machine Learning Techniques for Anomaly Detection in Pharmaceutical Quality Control

Gruppo scientifico disciplinare IINF/05-A

Dottorando

Dott. Niccolò Ferrari

Tutore

Prof.ssa Evelina Lamma

Cotutore

Dott. Davide Luisari

Anni 2021/2024

Abstract

The rapid advancements in Machine Learning (ML) and Computer Vision (CV) have revolutionized industrial quality control, especially in pharmaceutical manufacturing, where strict safety and regulatory standards demand exceptional product quality. Traditional quality control methods—largely reliant on manual inspection—are time-consuming, error-prone, and lack the scalability required for high-throughput production. This dissertation addresses these issues by developing ML-driven techniques for anomaly detection in pharmaceutical quality control, with a focus on high-speed industrial lines.

A primary challenge is detecting Out-of-Distribution (OOD) anomalies, namely defects that deviate from the standard production distribution. Traditional Computer Vision methods based on blob-analysis struggle with such anomalies because of their rigid, parameter-dependent nature. In contrast, modern deep learning approaches, which learn from extensive datasets, offer enhanced flexibility and scalability. However, deploying these models in real-time environments introduces further challenges, including dataset imbalance, strict inference time limits, and hardware constraints.

The chief contribution of this thesis is the development of the novel Generative-Reconstructive-Discriminative Network (GRD-Net), engineered to achieve both accurate anomaly detection and efficient real-time processing. GRD-Net exploits the capabilities of Generative Adversarial Networks (GANs) to reconstruct defect-free images while simultaneously generating heatmaps that highlight anomalous regions. This dual-stage process improves defect localization precision and overall system accuracy, marking a significant advancement over traditional methods—particularly in detecting small and irregular defects. Benchmarking on both public and proprietary pharmaceutical datasets confirms its robustness and efficiency in real-world conditions.

In parallel, the research introduces a derivative architecture, the Residual Generative Adversarial Network for Anomaly Detection (ResGANAD), specifically optimized for industrial deployment. Successfully integrated into high-speed Blow-Fill-Seal (BFS) vial inspection lines, ResGANAD detects OOD anomalies in real-time with minimal disruption to production throughput while maintaining high accuracy.

Furthermore, this dissertation evaluates and adapts embedding similarity-based techniques such as PaDiM and PatchCore, which prove effective in scenarios with scarce labeled anomaly data. Although these methods traditionally rely on memory-intensive components like memory banks—limiting their scalability—the thesis proposes adaptations that reduce dependence on pre-trained networks and enhance performance in large-scale industrial settings.

Extensive experimental results demonstrate the efficacy of the proposed models across a diverse range of real-world defects, including subtle anomalies that conventional inspection algorithms often overlook. Moreover, the models are rigorously tested against the stringent regulatory and safety requirements of pharmaceutical manufacturing, ensuring both high detection accuracy and real-time operational capability.

In summary, this dissertation advances the application of machine learning to anomaly detection in pharmaceutical quality control. The developed systems enhance the accuracy and speed of defect detection while offering scalable solutions suitable for high-throughput industrial environments. By integrating these advanced ML techniques, manufacturers can improve product quality, reduce reliance on manual inspection, and achieve higher operational efficiency. Future work will focus on further optimizing these models, adapting them to emerging industrial processes, and incorporating explainability mechanisms to strengthen trust and regulatory compliance.

Sommario

I rapidi progressi nel Machine Learning (ML) e nella Computer Vision (CV) hanno trasformato il controllo qualità industriale, specialmente nella produzione farmaceutica, dove i rigidi standard di sicurezza e le normative richiedono una qualità del prodotto eccezionale. I metodi tradizionali, basati sull'ispezione manuale, risultano lenti, soggetti a errori e non scalabili per produzioni ad alto rendimento. Questa dissertazione affronta tali problematiche sviluppando tecniche ML per il rilevamento di anomalie nel controllo qualità farmaceutico, con particolare attenzione alle linee industriali ad alta velocità.

Una sfida fondamentale è il rilevamento delle anomalie Out-of-Distribution (OOD), ovvero difetti che deviano dalla distribuzione standard di produzione. I metodi classici di CV, fondati sull'analisi di blob, faticano a gestire tali difformità a causa della loro rigidità e della dipendenza da parametri. Al contrario, gli approcci di deep learning, che apprendono da ampi dataset, offrono maggiore flessibilità e scalabilità, sebbene l'impiego in ambienti real-time comporti ulteriori difficoltà, quali squilibri nei dati, tempi di inferenza stringenti e vincoli hardware.

Il contributo principale di questa tesi è lo sviluppo della Generative-Reconstructive-Discriminative Network (GRD-Net), progettata per garantire un rilevamento accurato delle anomalie e un'elaborazione in tempo reale efficiente. GRD-Net sfrutta le capacità delle Generative Adversarial Networks (GANs) per ricostruire immagini prive di difetti, generando simultaneamente heatmap che evidenziano le regioni anomale. Questo processo in due fasi incrementa la precisione nella localizzazione dei difetti e l'accuratezza del sistema, rappresentando un significativo passo avanti rispetto ai metodi tradizionali, in particolare per l'individuazione di difetti piccoli e irregolari. I test effettuati su dataset farmaceutici, sia pubblici che proprietari, ne hanno confermato la robustezza in condizioni reali.

Parallelamente, la ricerca introduce la Residual Generative Adversarial Network for Anomaly Detection (ResGANAD), un'architettura derivata ottimizzata per l'uso industriale. Integrata con successo nelle linee di ispezione in tempo reale dei flaconi Blow-Fill-Seal (BFS) ad alta velocità, ResGA-

NAD rileva anomalie OOD in tempo reale, garantendo un impatto minimo sul throughput produttivo e mantenendo elevati standard di accuratezza.

La dissertazione valuta inoltre tecniche basate sulla similarità degli embedding, come PaDiM e PatchCore, efficaci in scenari con pochi dati etichettati. Sebbene tali metodi si fondino su componenti memory-intensive, come le memory bank, che ne limitano la scalabilità, si propongono adattamenti per ridurre la dipendenza dalle reti pre-addestrate e migliorare le prestazioni in ambienti industriali su larga scala.

Risultati sperimentali estesi dimostrano l'efficacia dei modelli proposti su una varietà di difetti reali, inclusi alcuni piccoli che spesso sfuggono agli algoritmi convenzionali. I modelli sono stati testati rigorosamente rispetto ai severi requisiti normativi e di sicurezza della produzione farmaceutica, assicurando elevata precisione e operatività in tempo reale.

In sintesi, questa dissertazione amplia l'applicazione del ML nel rilevamento di anomalie per il controllo qualità farmaceutico. I sistemi sviluppati migliorano la rapidità e l'accuratezza nell'individuazione dei difetti, offrendo soluzioni scalabili per ambienti industriali ad alto rendimento. L'integrazione di tali tecniche ML consente di elevare la qualità del prodotto, ridurre la dipendenza dall'ispezione manuale e aumentare l'efficienza operativa. Il lavoro futuro si concentrerà sull'ottimizzazione dei modelli, sull'adattabilità a nuovi processi industriali e sull'introduzione di meccanismi di explainability per rafforzare la fiducia e la conformità normativa.

Acknowledgements

First of all, I would like to thank my supervisor, Evelina Lamma, for guiding and supporting me throughout this journey and allowing me to explore many fascinating research topics. I am also grateful to Fabrizio Riguzzi for reading and providing valuable comments on my thesis during the review process.

I am grateful to my family—my parents Lucia and Andrea, my brother Giovanni, my aunt Carla, and my girlfriend Isabella—for their unconditional support in all my endeavors and activities. My heartfelt appreciation extends to my friends, for their support and for the wonderful moments of fun and care.

I extend my thanks to the University of Ferrara and Bonfiglioli Engineering for financing and supporting me throughout my academic journey, providing facilities, clusters, and data essential for advancing my studies. I am thankful to the entire R&D, engineering, and visual inspection teams: Sergio Dolci, Oligert Osmani, Federico Magni, Alessandro Di Cunzolo, Nicola Zanarini, as well as Davide Luisari and Davide Formenti, whose contributions were crucial for the initiation and continuation of this project.

Finally, I would like to express my gratitude to my friend Michele Fraccaroli, with whom I have shared studies and projects, and without whom many of these achievements would not have been possible.

Niccolò Ferrari

This work is dedicated to Giovanni and Isabella.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AD	Anomaly Detection
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
AnoSeg	Anomaly Segmentation Network
AUC	Area Under the Curve
AUROC	Area Under the Receiver Operating Characteristic
BFS	Blow Fill Seal
CNN	Convolutional Neural Network
CRAE	Convolutional bottleneck Residual Autoencoder
CRF	Conditional Random Field
CV	Computer Vision
CT	Computed Tomography
DCNN	Deep Convolutional Neural Network
DFN	Deep Feedforward Network
DL	Deep Learning
DNN	Deep Neural Network

DRAE Dense bottleneck Residual Autoencoder

DRÆM Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection

DT Decision Tree

ELU Exponential Linear Unit

EMA European Medicines Agency

FDA Food and Drug Administration

FFN Feed-Forward Network

FL Focal Loss

FLOPS Floating Point Operations per Second

FOL First-order Logic

GAN Generative Adversarial Network

GELU Gaussian Error Linear Unit

GMP Good Manufacturing Practice

GP Gaussian Process

GPT Generative Pre-trained Transformer

Grad-CAM Gradient-weighted Class Activation Mapping

GRD-Net Generative Reconstructive Discriminative Network

GRU Gated Recurrent Unit

HP Hyper-Parameter

k-NN k-Nearest Neighbours

Leaky-ReLU Leaky Rectified Linear Units

LLM Large Language Model

LSTM Long Short-Term Memory

MAE Mean absolute error

Masked-AE Masked autoencoder

Mask R-CNN Mask Region Based Convolutional Neural Networks

MhBC-PatchCore Mahalanobis-Based Clusters PatchCore

ML Machine Learning

MV Machine Vision

MLP Multilayer Perceptron

MSE Mean squared error

NDA Non Disclosure Agreement

NLP Natural Language Processing

NN Neural Network

OD Overlap Distance

OOD Out-Of-Distribution

OOD-AD Out-Of-Distribution Anomaly Detection

PCA Principal Component Analysis

PR Precision-Recall

QC Quality Control

ReLU Rectified Linear Units

ResAE Residual Autoencoder

ResGANAD Residual Generative Adversarial Network for Anomaly Detection

ResNet Residual Network

RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROI	Region of Interest
SELU	Scaled Exponential Linear Unit
SiLU	Sigmoid Linear Unit
SOTA	State-of-the-Art
SSIM	Structural Similarity Index
SRP	Sparse Random Projection
SVM	Support-Vector Machine
tanh	hyperbolic tangent
VAE	Variational Autoencoder
VI	Visual Inspection
ViT	Vision Transformer
VQ-VAE	Vector Quantized Variational Autoencoder
WGAN	Wasserstein Generative Adversarial Network
WRAE	Wide Residual Autoencoder
WRN	Wide Residual Network

Part I

Introduction

Chapter 1

Motivation and Goal of the Thesis

1.1 Motivation

Anomaly Detection (AD) is becoming increasingly crucial in today's industrial context, where quality control at the conclusion of the manufacturing process is one of the most significant stages. Today, manual operators frequently address this extensively or in samples [43, 44, 146, 9]. In certain sectors, quality is not always associated with strong business standards, but it does, above all, protect consumer safety, particularly in the medical and pharmaceutical industries, where quality control has a direct impact on health [120]. In the pharmaceutical industry, human operators still perform inline checks on packages, but they confront clear challenges, such as the danger of spotting a flaw, as well as low efficiency, which limits production.

Automatic algorithms, on the other hand, can run inline high-speed controls because of their lower error rate, enhanced systematicity, and higher throughput. Traditional techniques for Visual Inspection (VI) Computer Vision (CV) rely on blob-analysis algorithms, which are less flexible because of their huge number of parameters and less capable of generalization due to their intrinsic rigidity. Nowadays, thanks to continuous hardware development in terms of performance and compactness, as well as the academic formalization of increasingly sophisticated Machine Learning (ML) models, the industry has begun to shift toward automatic learning technologies based on Deep Learning (DL) algorithms. Despite the potential advantages of machine DL algo-

rithms, there exist critical considerations that must not be overlooked. One of the primary concerns, particularly within the pharmaceutical sector, is the explainability of the results derived from these algorithms. Being able to clearly explain these results is really important. This is crucial as it allows for better understanding and trust in the model's predictions.

The initial technologies that emerged were related to supervised learning. Although the number of samples for each class defined a priori has been reduced thanks to deep convolutional models of automatic learning, there were still several obstacles to the use of such technology in this field. Generally, industrial lines are capable of producing a large quantity of conforming products. However, to create a library of real anomalies, it could take a lot of time and only after actual production. The alternative is to construct a plausible library of flaws, which may not cover all possible scenarios. Furthermore, producing as many samples as there are good ones could be incredibly expensive and time-consuming. For these reasons, supervised learning algorithms are not commonly employed in real-world production inspections.

Semi-supervised algorithms have been lately applied with the purpose of learning the distribution of the features of a known label, often those of positive products, which are in higher quantity and easier to retrieve. The primary task has shifted from classifying inputs into predetermined labels. The focus now lies on identifying anomalies, specifically outliers, by performing Out-Of-Distribution Anomaly Detection (OOD-AD).

1.2 Goal of the Thesis

The goal of this thesis is to provide new Anomaly Detection techniques and relative explicability of the result obtained, to be integrated in high-speed and critical industrial production lines.

First proposed is a novel two-step architecture, designed with the capability to reconstruct an input image devoid of any potential anomalies. The primary objective of this architecture is to identify these anomalies and subsequently generate a corresponding heat map. Upon the generation of this heat map, a mask can be effortlessly derived, which can then be utilized to segment

the source image. This segmentation aids in pinpointing the exact location of the anomaly within the image. The model, aptly named the GRD-Net [43], leverages the architecture of GANs. The primary function of the GRD-Net is to learn the reconstruction of the input image, which may have a random anomaly patched onto it. The Generative component of the network strives to rebuild the original image, effectively eliminating the superimposed noise. Concurrently, the Discriminative component generates the heat map, utilizing both the original and the rebuilt image as sources. This comprehensive approach allows for a more accurate and detailed identification and analysis of anomalies within images.

The second system proposed in this dissertation is an innovative architecture, which is a derivative of the previously outlined GRD-Net model. This architecture, named Residual Generative Adversarial Network for Anomaly Detection (ResGANAD) [44], has been readapted and integrated into a high-speed industrial production line, specifically designed for visual inspection and quality control systems. The primary design objective of this network is to detect Out-Of-Distribution (OOD) anomalies. The application of this network is particularly relevant in the pharmaceutical industry, where it is employed to inspect BFS plastic strips of vials. The network's advanced anomaly detection capabilities ensure the highest standards of Quality Control (QC). This significantly enhances not only the overall efficiency and reliability of the pharmaceutical manufacturing process, but also maintains a balance with high throughput. This ensures that the process is optimized without compromising on the throughput of the production line.

The last proposed contribution is a review of architectures for anomaly detection based on the concept of embedding similarity. These systems have as their main strength the need for a very small number of positive examples for training and a reduced test set to train the last threshold and discern between good and anomaly. Their realization in the case of large and complex training sets is often not so trivial, due to the fact that they are often based on the concept of "*memory bank*" which grows exponentially with the increase of the training set. Furthermore, being based on pre-trained networks, usually residual networks like ResNet18, ResNet50 or Wide-ResNet50 trained on ImageNet or a subsequent version, they contain an intrinsic bias that is different

from the specific one of the application case. Therefore, architectures based on the concept of reference ones, such as PaDiM, PatchCore, PNI, etc., will be proposed, trying to overcome the limits based on the pre-training domain and approximate the result, in order to address the problem of the *memory bank* and its inapplicability in the case of very large datasets, as mentioned above.

Chapter 2

Structure of the Thesis

This thesis is divided into five parts:

- The first part introduces the motivations and goal of the thesis.
- The second part introduces the background concepts of Deep Learning, Machine Vision and Anomaly Detection in Machine Learning.
- The third part discusses the concept of Anomaly Detection algorithms and systems created during the PhD programme and their applications in the industrial context at Bonfiglioli Engineering.
- The fourth part introduces the industrial applications of the methods described in part three and the results obtained in benchmarking and real-cases pharmaceutical datasets.
- The fifth part concludes this dissertation with several possible directions and future work.

The Deep Learning chapter of the second part summarizes concepts extrapolated from:

- Goodfellow, I., Bengio, Y., & Courville, A. (2016), Deep Learning, MIT Press. [55]
- Niccolò Ferrari, Michele Fraccaroli, Evelina Lamma, GRD-Net: Generative-Reconstructive-Discriminative Anomaly Detection with Region of Interest Attention Module, International Journal of Intelligent Systems, Wiley, 2023. [43]

- Riccardo Zese, Elena Bellodi, Michele Fraccaroli, Fabrizio Riguzzi, and Evelina Lamma. Neural Networks and Deep Learning Fundamentals, pages 23–42. Springer International Publishing, Cham, 2022. [171]

The third part is described in an exhaustive and in-depth way in this publication:

- Niccolò Ferrari, Michele Fraccaroli, Evelina Lamma GRD-Net: Generative-Reconstructive-Discriminative Anomaly Detection with Region of Interest Attention Module, International Journal of Intelligent Systems, Wiley, 2023. [43]

This chapter also provides an in-depth description of sophisticated techniques used within the industrial field for performing anomaly detection and defect segmentation on real datasets. The experiments and application of the fourth part are described more extensively also in this publication:

- Ferrari Niccolò, Zanarini Nicola, Fraccaroli Michele, Bizzarri Alice, Lamma Evelina, Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line. Available at SSRN: https://ssrn.com/abstract=4858664 or http://dx.doi.org/10.2139/ssrn.4858664. [44]

2.1 How to Read this Thesis

We tried to make this work as understandable and logically smooth as possible. The main topic of this work is the *Anomaly Detection (AD)* of Deep Learning (DL) models. Following the Figure 2.1, Chapters 3 to 5 compose the background. Chapters 6 to 8 are important for correctly understanding the part of the thesis that compose the left branch of Figure 2.1. Chapters 9 to 10 compose the integration in real-world industrial processes.

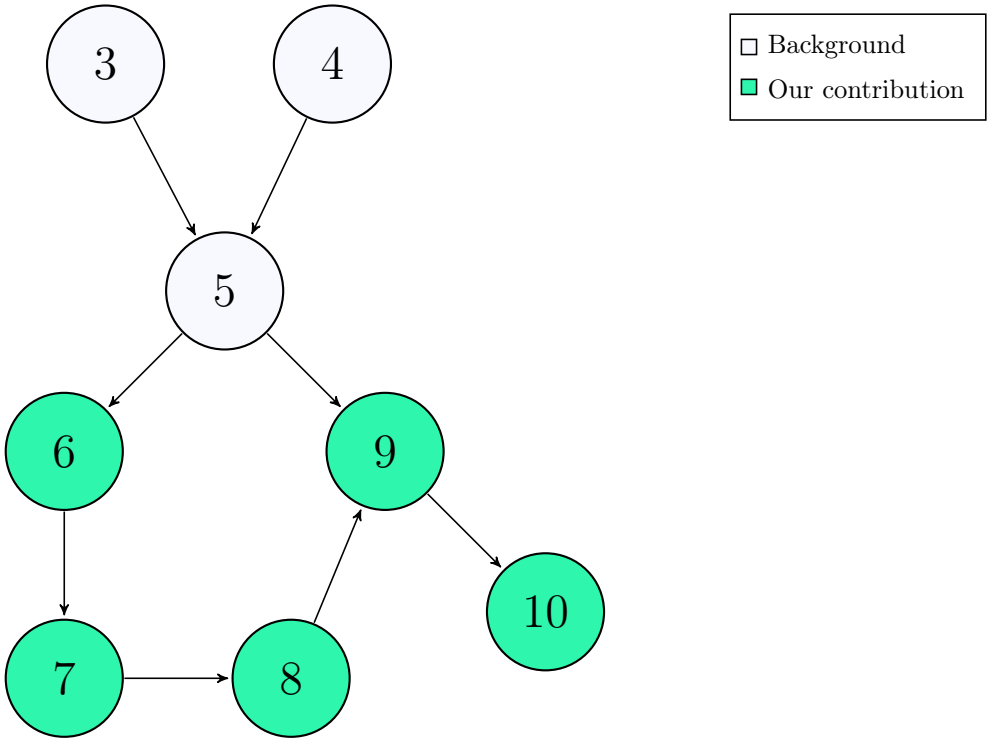


Figure 2.1: Chapter dependency graph. For instance, to understand Chapter 9, you have to read Chapter 5 and 8.

Part II

Background

Chapter 3

Deep Learning

Deep Learning (DL) is a Machine Learning (ML) technique inspired by the principles of neuroscience. It employs a computational model known as Artificial Neural Network (ANN), which is derived from the structure and functioning of biological neural networks [106]. DL utilizes these computational models, referred to as neural networks, which are composed of multiple hierarchical levels. This architecture enables the learning of data representations with various levels of abstraction, where each level builds upon the concepts learned at previous levels. The process of stacking these layers and increasing the number of units (neurons) within each layer allows neural networks to represent functions with escalating complexity.

Different DL architectures, including Deep Feedforward Network (DFN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Transformer (such as Vision Transformer (ViT)), have found applications across a wide range of fields. These include computer vision, where they are used for image and video analysis; speech and audio recognition, which involves processing and interpreting auditory data; natural language processing, which focuses on understanding and generating human language or generating data, such as images or videos, from a given prompt; social network filtering, which helps in managing and curating content; bioinformatics, where they assist in understanding biological data; and numerous other sectors.

These systems learn primarily in two ways: *supervised* and *unsupervised* learning.

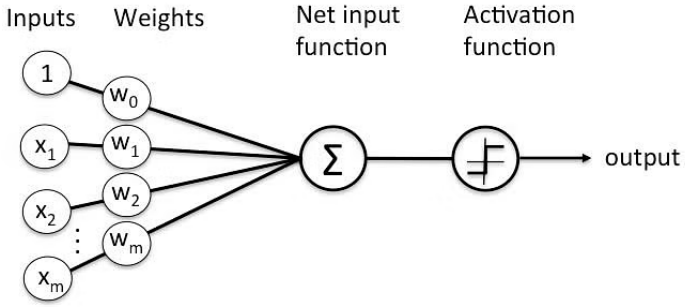


Figure 3.1: Diagram illustrating the structure of an artificial neuron (perceptron).

Supervised Learning is a method that relies on labeled examples provided during the training phase. Each example consists of an input object paired with a desired output value (label). This output value represents the target that the network aims to predict. In *Supervised Learning*, it is crucial to supply the network with a sufficient amount of labeled examples. Using this data, the network constructs an *interpretation* of the input, which enables it to accurately classify the provided examples. Essentially, the network analyzes the given data and develops an inferred function that can be used to map new, unseen examples to their correct outputs.

Unsupervised Learning, on the other hand, operates differently as the examples provided to the network do not include output labels. In this approach, the network uses the input data to learn common features and patterns, which it can then use to make predictions or decisions about future inputs.

A key example of a deep learning model is the *Multilayer Perceptron (MLP)* [55], which builds upon the *perceptron* concept [117]. An MLP consists of at least three types of layers: an input layer, one or more hidden layers, and an output layer. In this network, neurons (excluding those in the input layer) use non-linear activation functions to determine their outputs. These activation functions are critical as they generate the output of each neuron based on its inputs. This output then becomes the input for the neurons in the next layer, allowing the network to progressively process and modify the data through multiple layers.

Figure 3.1 shows a diagrammatic representation of the structure of an artificial neuron, also known as a perceptron.

3.1 Deep Feed-forward Networks

Deep Feedforward Networks (DFNs), often referred to as Multilayer Perceptrons (MLPs), consist of multiple layers between the input and output layers. These networks define a function mapping $y = f(X, \theta)$, where X represents the input data, y denotes the predicted output or category, and θ signifies the parameters that the network needs to learn [55]. The objective is to approximate a function $f : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^k$, expressed as $f(X) = f_n(f_{n-1}(...(f_1(f_0(X)))))$. Here, X is the input tensor (a multi-dimensional array with n dimensions), and the network comprises n layers, each performing a function f_j, where j denotes the layer index.

3.1.1 Architecture and Function

In a DFN, data flows from the input X to the output Y in a one-way, forward direction, without any feedback loops. This feed-forward mechanism is crucial for many deep learning applications. DFNs form the backbone of more complex network structures, such as Recurrent Neural Networks (RNNs) [145] and Convolutional Neural Networks (CNNs). In particular, CNNs are a fundamental network type used widely in various applications, and their principles are essential to the techniques explored in this study.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are inspired by biological processes [48], where the connection between neurons is similar to that of the animal visual cortex: every neuron responds to stimuli in a specific region of the visual field, the receptive field. Receptive fields partially overlap and cover the entire visual field.

Convolutional Neural Network (CNN) are the primary network architecture for image analysis. *Convolution* is a mathematical operation between two functions of a variable, defined as the integration of products between the first function and the second translated by a certain value:

3.2.1 Mathematical Foundation

The following equation (Eq. 3.1) defines convolution between continuous functions:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(a)g(t-a)da \tag{3.1}$$

while equation (Eq. 3.2) adapts it to discrete data;

$$(f * g)[n] = \sum_{a=-\infty}^{\infty} f(a)g(n-a) \tag{3.2}$$

.

Moving from mathematics to CNNs, the function f is the *input*, the function g is the *kernel*, and the output is called the *feature map*.

3.2.2 Network Design

The architecture of CNNs consists of an input layer, an output layer, and multiple hidden layers. These hidden layers typically include:

- Convolutional layers
- Activation functions
- Pooling layers

3.2.2.1 Convolutional Layer

The **convolutional layer** is parameterized by a tensor with shape *(image width)* x *(image height)* x *(image depth)* (or channels). Image depth is the number of color channels (for color images, channels are 3: red, green, and blue). Given an input matrix, e.g., an image, this layer performs filtering with a filter called a *kernel*. The region of the matrix (or image) covered by a kernel is called the **receptive field**. The filtering operation consists of multiplying the filter values with the pixel values of the input tensor, generating a unique number in the output that represents the single receptive field. This operation is repeated, moving the filter over the whole image by a predefined unit. At the end of this operation, we have the *feature map* as output, a smaller matrix

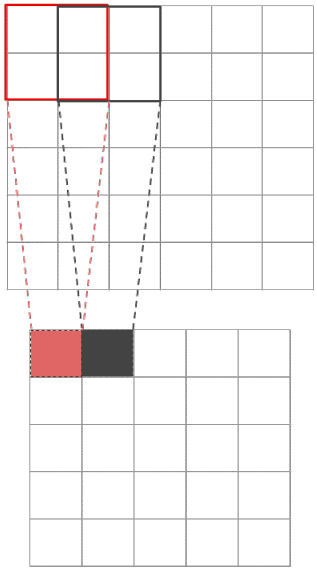


Figure 3.2: Input tensor (top grid) with kernel 2x2 (red square) in the first position and the respective point or result on feature map (bottom grid). The grey square represents the second step of filtering.

than the input that represents an extraction of significant features found in the original image. The graphical situation is represented in Figure 3.2.

3.2.2.2 Activation Layer

Following the convolutional layer, there is a non-linear layer or *activation layer*. The purpose of this layer is to introduce non-linearity. There are different types of activation functions, such as sigmoid or tanh, but the best choice for hidden layers is the *Rectified Linear Units (ReLU)* [115, 55]. The principal functions are described in Figures 3.3 and 3.4. ReLU and Exponential Linear Unit (ELU) are preferred because they shorten the network's training time while maintaining good accuracy. The ELU function, unlike ReLU, tries to make the mean activations closer to zero. ELU can achieve higher classification accuracy than ReLU [24]. There are also various activation functions available, each with its own characteristics and applications in neural networks, they can be summarized as following:

- **Rectified Linear Units (ReLU)**: The ReLU function is defined as $f(x) = \max(0, x)$. It is widely used due to its simplicity and effective-

ness in reducing the training time and preventing the vanishing gradient problem [115, 55].

- **Exponential Linear Unit (ELU)**: The ELU function includes a small curve when the input is negative, which helps to push the mean activations closer to zero and improves the training dynamics. ELU can achieve higher classification accuracy than ReLU. Mathematical function is defined as:

$$y = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases} \tag{3.3}$$

  , where the parameter α is a hyper-parameter to be tuned. [24].

- **Scaled Exponential Linear Unit (SELU)**: This function automatically scales the outputs to have zero mean and unit variance, aiming to make the network self-normalizing [87].

- **Gaussian Error Linear Unit (GELU)**: This function approximates the outputs of a neuron to be Gaussian-distributed, which can be beneficial for transformer models and some convolutional neural networks [68].

- **Leaky Rectified Linear Units (Leaky-ReLU)**: It modifies ReLU by allowing a small, positive gradient when the unit is inactive, defined as $f(x) = \max(0.01x, x)$. It helps to keep the gradient flow alive during the training process [105].

- **Sigmoid Linear Unit (SiLU)**: This function is defined as $f(x) = x \cdot \sigma(x)$, where $\sigma(x)$ is the sigmoid function. SiLU has been shown to outperform ReLU in deeper networks [39].

- **Swish**: A variant of SiLU proposed by researchers at Google, which has shown potential in various tests over traditional ReLU [129].

Figures 3.3 and 3.4 describe these functions graphically.

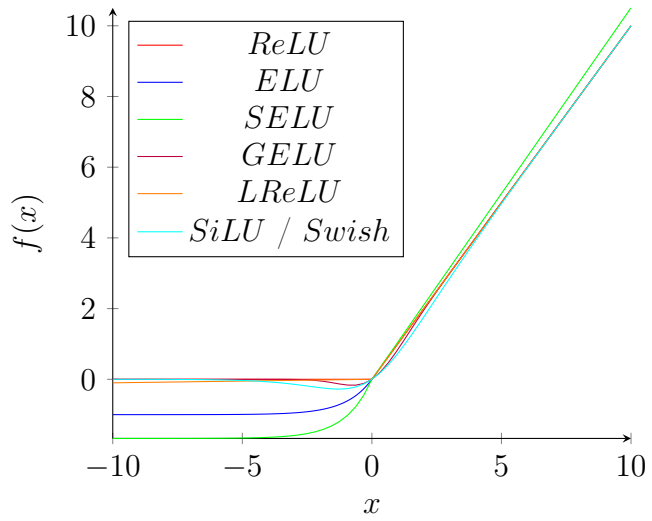


Figure 3.3: Activation functions: ReLU, ELU, SELU, GELU, Leaky ReLU, and SiLU/Swish.

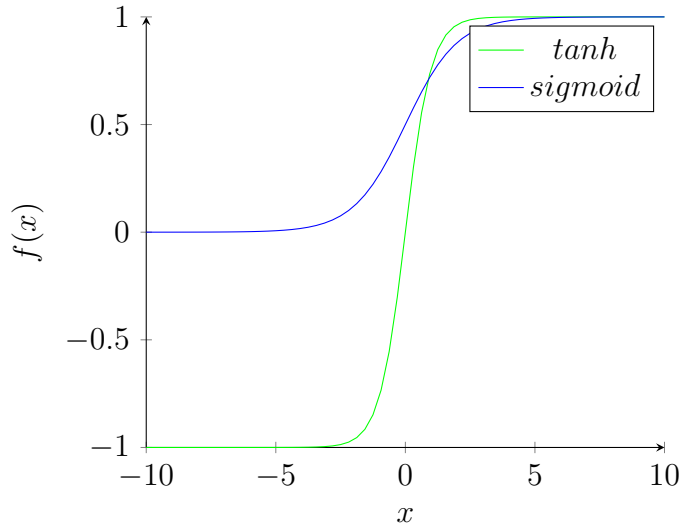


Figure 3.4: Sigmoid and tanh functions.

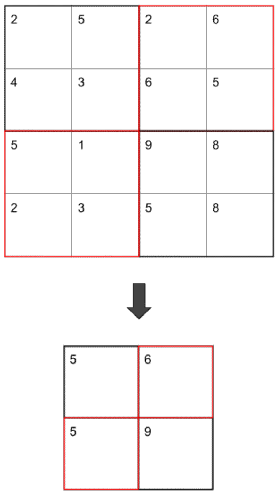


Figure 3.5: This is an example of max-pooling. The output of the convolutional layer (top grid) is the input of the pooling layer with kernel 2x2 and stride 2, and the bottom grid is the result of max-pooling.

3.2.2.3 Pooling Layer

Following the activation layer is the *pooling layer*. Pooling can be global or local. This layer reduces the dimensions of the matrix using a kernel and a stride of the same length. Global pooling acts on all the neurons of the previous convolutional layer, while local pooling combines small clusters, as shown in Figure 3.5. The operation applied by the pooling layer can be a max or an average. Max-pooling uses the maximum value from each cluster, while average pooling uses the average value from each cluster. For each filtering operation, the layer maintains the highest (or the mean) value of parameters, thereby obtaining a reduced matrix. Another purpose of pooling is to control overfitting, as many values are deleted, keeping only the most significant ones.

3.2.2.4 Fully Connected Layer

The final layer, and also the output layer of this network, is the **fully connected** layer. This layer connects every neuron of a layer to every neuron of the layer above. This layer is used for classification and returns the final output of the entire CNN, as shown in Figure 3.6.

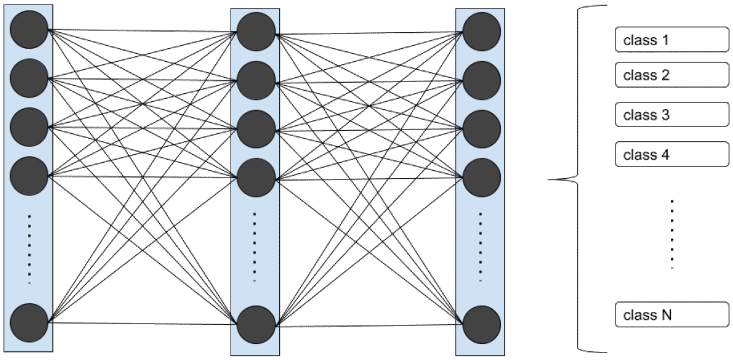


Figure 3.6: Some layers of fully connected neurons terminating in classification.

3.3 Autoencoders

An AE [55, 7, 110] is a type of neural network that is trained to try to replicate its input to its output. This network can be decomposed into two parts: an encoder (E) that compresses the input into latent space h and a decoder (D) that produces a reconstruction of the input starting from the latent space h as shown in Figure 3.7. Then, if x is the input given to the network, the E can be defined as $h = f(x)$ where f is the encoder function, D can be defined as $r = g(h)$ where g is the decoder function and r is the reconstruction of x. Then, an AE learns to set $g(h) \rightarrow g(f(x)) = x$.

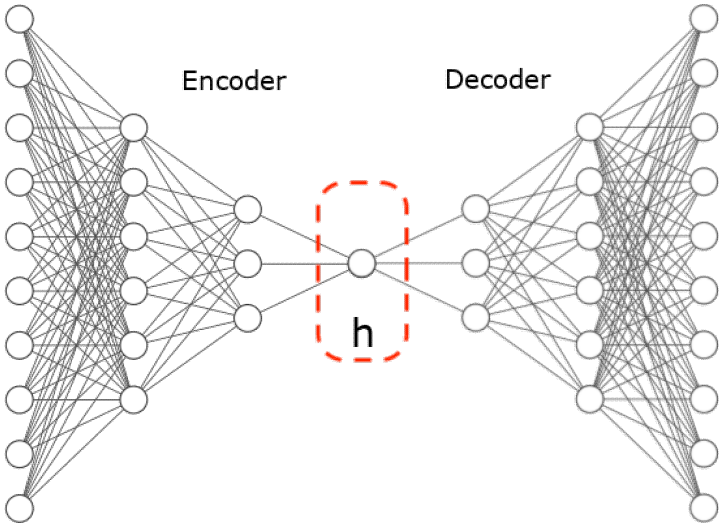


Figure 3.7: Representation of AE.

The potential of AE lies in the input copying task and in the possibility of constraining h to be smaller than x (in this case, we talk about *undercomplete* AE). Learning an undercomplete representation forces the network to identify the most important features of the input data. This process can be performed by minimizing the function $l(x, g(f(x)))$ where l is the loss function that penalizes the network when $g(f(x))$ is distant from x.

AEs are the simplest form of AE. They consist of an encoder and a decoder with symmetric architectures. These models are trained to minimize the reconstruction error between the input and the output. Despite their simplicity, classical AEs have been widely used for dimensionality reduction and feature learning.

3.3.1 Sparse Autoencoders

A Sparse AE imposes a sparsity constraint on the latent space as a training criterion. This type of AE is used to enforce the network to learn a compact and efficient representation of the input data, which can be useful for tasks like classification.

3.3.2 Denoising Autoencoders

Denoising AEs are trained to remove noise from the input data. Instead of applying a sparsity penalty, the network minimizes the function $l(x, g(f(\tilde{x})))$ where $\tilde{x}$ is a noisy version of x. This helps the AE learn robust features that are resilient to noise.

3.3.3 Variational Autoencoders

A Variational Autoencoder (VAE) [86, 84] is a generative model that learns to encode inputs as distributions over the latent space rather than fixed points. The encoder defines an approximate posterior distribution $q(h|x)$, and the decoder defines the conditional distribution $p(x|h)$. This allows the model to generate new data by sampling from the latent space distribution.

3.3.4 Vector Quantization in VAEs

Vector Quantized Variational Autoencoder (VQ-VAE) [157] introduces discrete latent variables. Instead of continuous latent space, VQ-VAE maps inputs to the nearest point in a fixed set of embeddings, which enables the use of discrete latent representations and combines the advantages of both AEs and quantization techniques.

3.3.5 Masked Autoencoders

Masked AEs are designed to handle missing data or to impute missing values in the input data. These models learn to reconstruct the full input from partially observed inputs by predicting the missing values, which is particularly useful in tasks like data cleaning and imputation.

3.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of machine learning frameworks introduced by Ian Goodfellow and his colleagues in 2014 [56]. They consist of two neural networks, the generator and the discriminator, which are trained simultaneously through adversarial processes.

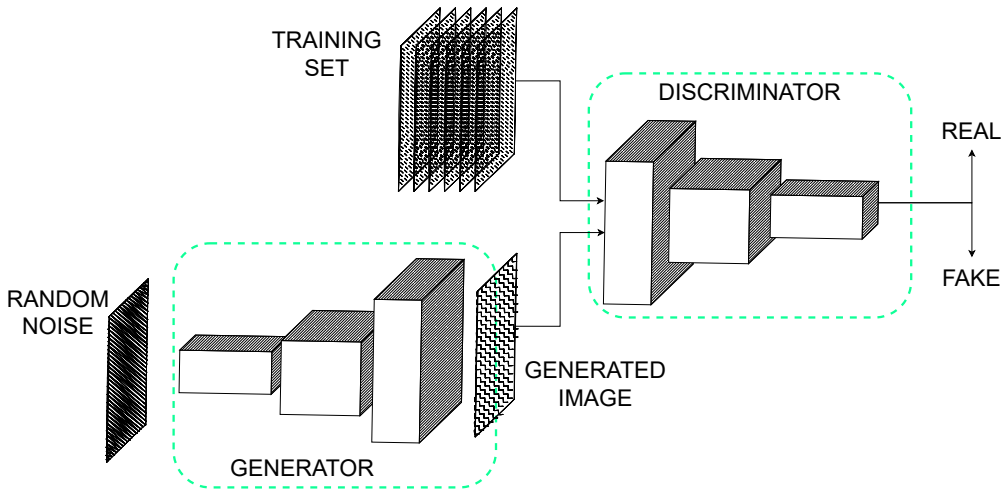


Figure 3.8: Representation of GAN.

3.4.1 Architecture

The architecture of GANs includes two main components:

- **Generator**: The generator G creates data samples from random noise z, aiming to produce outputs that resemble the training data.
- **Discriminator**: The discriminator D evaluates the authenticity of the samples, distinguishing between real data and those generated by G.

as shown in Figure 3.8.

The two networks are trained in a zero-sum game, where the generator aims to maximize the probability of the discriminator making a mistake, while the discriminator aims to minimize its own error.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (3.4)$$

3.4.2 Training Process

The training process of GANs involves the following steps:

1. The generator produces a batch of fake data samples from random noise.
2. The discriminator evaluates the fake data along with a batch of real data samples.

3. The discriminator is trained to distinguish real from fake data.

4. The generator is updated to produce more realistic data, using the feedback from the discriminator.

3.4.3 Applications and Variants

GANs have found applications in various domains, including:

- **Image Generation and Style Transfer**: Creating realistic images, as demonstrated by Progressive GAN [80] and StyleGAN [82].
- **Data Augmentation**: Enhancing datasets for improved training of other models.
- **Image-to-Image Translation**: Converting images from one domain to another, such as turning sketches into photos [175].

Several variants of GANs have been proposed to address specific challenges, such as training instability and mode collapse. Notable examples include:

- **Progressive GANs**: Improve the quality of generated images by progressively growing the network during training [80].
- **StyleGAN**: Allows for fine-grained control over the generated images by incorporating style-based layers [82].
- **CycleGAN**: Facilitates image-to-image translation between unpaired datasets [175].
- **Wasserstein Generative Adversarial Network (WGAN)**: Enhances training stability and reduces mode collapse by using the Wasserstein distance [4].

3.4.3.1 WGAN

WGAN is a variant of GAN that addresses issues related to training stability and mode collapse in traditional GANs. Instead of using the Jensen-Shannon

divergence to measure the difference between the real and generated data distributions, WGAN uses the Wasserstein distance, which provides a smoother and more stable training process [4]. The loss function for WGAN is given by:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)}[D(x)] - \mathbb{E}_{z \sim p_z(z)}[D(G(z))] \tag{3.5}$$

where D is the critic function, which approximates the Wasserstein distance between the real and generated distributions. This approach helps in improving the quality of generated samples and achieving more stable convergence during training.

3.4.4 Anomaly Detection with GANs

GANs have been effectively utilized in AD tasks, leveraging their ability to generate data, which enhances the reconstruction capabilities of AEs as discussed in Section 3.3. This integration has led to the development of several sophisticated infrastructures based on GANs that are well-documented in the literature. Notable examples include AnoGAN [142], BiGAN [36], VAE-GAN [89], GANomaly [2], and its variant Skip-GANomaly [3].

3.4.4.1 GANomaly

One of the most noteworthy architectures is GANomaly, a specialized type of GAN designed specifically for anomaly detection. GANomaly innovatively combines the principles of AE and GAN to identify anomalies by learning to generate normal samples and recognizing deviations from these norms. In this architecture, the generator is employed to produce normal samples, while the discriminator's role is to differentiate between normal and abnormal samples. This dual mechanism enhances the effectiveness of GANomaly in various applications, including fraud detection and fault diagnosis [2, 3].

3.5 Diffusion Models

Diffusion models are a type of generative model that has gained significant attention for their ability to produce high-quality samples, such as images.

These models are inspired by the physical process of diffusion, where particles spread out over time, and they operate by reversing a process of adding noise [151, 72].

3.5.1 Overview

The operation of diffusion models can be divided into two main phases: the forward diffusion and the reverse diffusion [151]. In the forward diffusion process, data is progressively corrupted by the addition of Gaussian noise over a series of time steps, transforming the data distribution into a noise distribution. This transformation is modeled as a Markov chain, where each step incrementally adds a small amount of noise [151, 152]. Conversely, the reverse diffusion process aims to recover the original data from these noisy samples by learning a reverse Markov chain that denoises the data step-by-step [72, 152].

3.5.2 Key Components

Several key components are crucial for the functioning of diffusion models. The noise schedule is a critical aspect that defines the amount of noise added at each time step in the forward process, directly influencing the complexity of the denoising task in the reverse process [72]. The score network, another fundamental component, is a neural network trained to estimate the gradient of the data log-likelihood with respect to the noisy data, guiding the denoising process [152]. Additionally, the denoising network learns to predict clean data from its noisy inputs, effectively approximating the reverse diffusion process [72].

3.5.3 Training

Training diffusion models involves generating noisy versions of data samples through the application of Gaussian noise, known as the forward process [72]. The model then learns to reverse this noise addition during what is known as the reverse process, aiming to recover the original, uncorrupted data [152]. The training typically employs a loss function that could include reconstruction loss or likelihood-based loss, quantifying the model's efficacy in removing noise and recovering data.

3.5.4 Applications and Popular Diffusion Models

Diffusion models are applied in various fields including image and video generation, image denoising, and text-to-image synthesis. They are particularly noted for their ability to produce high-quality images from random noise, remove noise to restore image quality, generate images based on textual descriptions, and create video frames or sequences from noise [72, 152, 130, 73]. Some of the notable models in this domain include the Denoising Diffusion Probabilistic Models (DDPM) by Ho et al. [72], Score-Based Generative Models by Song et al. [152], and subsequent improvements that have enhanced sample quality and computational [118].

3.5.5 Mathematical Formulation

The diffusion process in diffusion models can be comprehensively described through its two principal components, each framed around the notions of Markov processes and KL-divergence, to model the data's transition through noisy states.

3.5.5.1 Markov Chains and Forward Process

The forward diffusion process is represented as a Markov chain, where the transition probability:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{3.6}$$

models the incremental addition of Gaussian noise at each time step t [72, 151]. Here, β_t denotes the variance schedule that controls the noise amount added at each step, and $\mathcal{N}$ signifies a Gaussian distribution. This forward process gradually transforms the clean data distribution into a latent noise distribution over T steps.

3.5.5.2 Reverse Process Transition Probability

The reverse process, aimed at recovering the original data from the noisy samples, uses the transition probability:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{3.7}$$

where μ_θ and Σ_θ represent the mean and covariance predicted by a neural network parameterized by θ [152, 118]. The neural network learns these parameters to effectively denoise the data, reversing the forward process.

3.5.5.3 Variational Lower Bound and Training Objective

Training diffusion models involves optimizing a variational lower bound (VLB) on the data likelihood:

$$\mathcal{L} = \mathbb{E}_q \left[\sum_{t=1}^{T} D_{KL} \left(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t) \right) - \log p_\theta(x_0|x_1) \right] \tag{3.8}$$

This expression includes the Kullback-Leibler divergence D_{KL}, which measures the efficiency of the reverse process in approximating the forward process, and the term $\log p_\theta(x_0|x_1)$ represents the reconstruction loss, pushing the model to generate outputs that closely mimic the true data [72, 152, 118].

3.5.5.4 Score Matching

In addition to traditional diffusion models, score-based generative models employ score matching techniques, training the model to estimate the gradient of the data distribution's log-density with respect to the noisy data:

$$\nabla_{x_t} \log p(x_t) \tag{3.9}$$

This score function is predicted using a neural network, which helps guide the reverse diffusion process [152, 72]. Score-based models focus on directly approximating these gradients, offering a different approach to learning the reverse transition probabilities.

Overall, the mathematical formulations of diffusion models employ a blend of stochastic processes and neural network optimization to effectively model

and reverse complex data transformations, making them powerful tools for generative modeling across various applications.

3.5.6 Advantages and Challenges

While diffusion models offer the advantage of generating high-quality samples and effectively modeling complex data distributions, they are not without challenges. These models are computationally intensive, requiring iterative denoising steps and careful tuning of noise schedules and training parameters [152].

In conclusion, diffusion models represent a significant advancement in generative modeling, providing a robust framework for creating realistic data samples across various domains.

3.6 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed for processing sequential data by maintaining a hidden state that captures information about previous elements in the sequence. Unlike traditional feed-forward networks, RNNs are well-suited for tasks where context and order are important [40].

3.6.1 Basic RNN Structure

An RNN processes an input sequence $X = [x_1, x_2, \ldots, x_T]$ by iterating through the sequence and updating its hidden state h_t at each time step t. The hidden state is updated based on the current input x_t and the previous hidden state h_{t-1}:

$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h) \tag{3.10}$$

where W_h and U_h are weight matrices, b_h is a bias vector, and σ is an activation function such as ReLU or hyperbolic tangent (tanh).

3.6.2 Challenges with RNNs

Despite their ability to handle sequential data, RNNs face several challenges:

- **Vanishing and Exploding Gradients**: During backpropagation, gradients can become very small or very large, making training difficult [11].
- **Long-Range Dependencies**: RNNs struggle to capture dependencies between distant elements in a sequence due to the iterative nature of the hidden state updates [74].

3.6.3 LSTM and GRU

To address these challenges, more advanced variants of RNNs have been developed, such as Long Short-Term Memorys (LSTMs) and Gated Recurrent Units (GRUs). These models include gating mechanisms to better control the flow of information and mitigate the vanishing gradient problem.

3.6.3.1 LSTM

LSTMs introduce memory cells and gates (input, output, and forget gates) to manage information flow:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.11}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.12}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.13}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.14}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.15}$$

3.6.3.2 GRU

GRUs simplify the gating mechanism by combining the forget and input gates into an update gate and merging the cell state and hidden state [23]:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{3.16}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{3.17}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3.18}$$

These improvements help LSTMs and GRUs capture longer-range dependencies more effectively than standard RNNs.

3.7 Attention

Attention mechanisms have become fundamental in many neural network architectures, enabling models to focus on specific parts of the input sequence when producing each element of the output sequence. This section provides an overview of the attention mechanism, including self-attention and multi-head attention.

3.7.1 Self-Attention Mechanism

The self-attention mechanism computes a representation of a sequence by relating each element to every other element, effectively capturing dependencies within the sequence [159]. Given a sequence of vectors $X = [x_1, x_2, \ldots, x_n]$,

self-attention produces an output sequence $Y = [y_1, y_2, \ldots, y_n]$ where each y_i is calculated as:

$$y_i = \sum_{j=1}^{n} \alpha_{ij} \cdot x_j \tag{3.19}$$

The attention weights α_{ij} are computed using a softmax function over the similarity scores, typically derived from scaled dot-product attention:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{3.20}$$

$$e_{ij} = \frac{(x_i W_Q)(x_j W_K)^T}{\sqrt{d_k}} \tag{3.21}$$

where W_Q and W_K are learnable weight matrices for the query and key projections, respectively, and d_k is the dimensionality of the key vectors.

3.7.2 Multi-Head Attention

Multi-head attention extends the self-attention mechanism by allowing the model to jointly attend to information from different representation subspaces at different positions [159]. Instead of a single attention function, multiple sets of attention weights are computed in parallel, each with its own parameters. These are concatenated and linearly transformed to produce the final output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h) W_O \tag{3.22}$$

$$\text{where head}_i = \text{Attention}(QW_{Q_i}, KW_{K_i}, VW_{V_i}) \tag{3.23}$$

Multi-head attention allows the model to capture different aspects of the input sequence relationships, enhancing its representational capacity.

3.8 Transformers

The Transformer architecture, introduced by Vaswani et al. in 2017 [159], revolutionized sequence modeling by entirely dispensing with recurrent and convolutional structures, relying solely on self-attention mechanisms to draw global dependencies between input and output.

3.8.1 Architecture Overview

A Transformer model consists of an encoder and a decoder, each composed of a stack of identical layers. Each layer in the encoder has two main sub-layers: multi-head self-attention and a position-wise fully connected feed-forward network (Feed-Forward Network (FFN)). The decoder layers have an additional multi-head attention sub-layer that attends to the encoder's output.

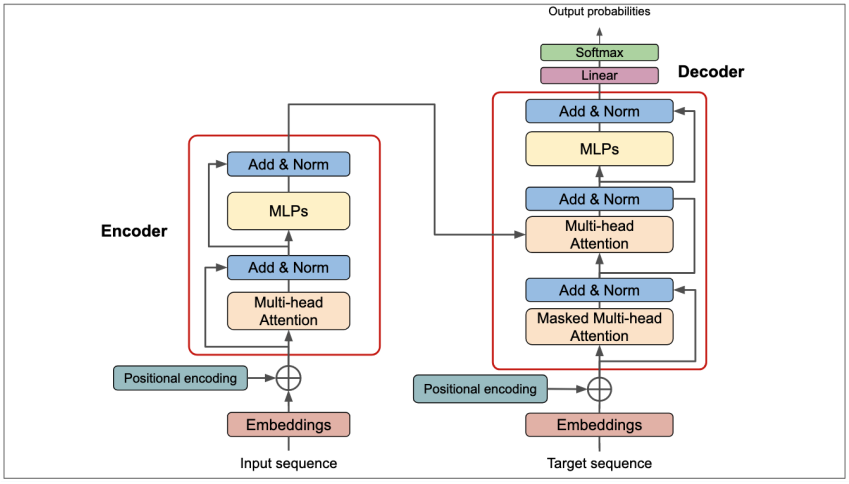


Figure 3.9: Transformer architecture with encoder and decoder stacks.

3.8.1.1 Positional Encoding

Since the Transformer lacks the sequential nature of RNNs, positional encodings are added to the input embeddings to provide information about the position of each token in the sequence:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3.24}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3.25}$$

These encodings are added to the input embeddings to incorporate positional information.

3.8.2 Advantages of Transformers

The Transformer architecture offers several advantages over traditional models:

- **Parallelization**: Unlike RNNs, Transformers allow for parallel computation of all elements in a sequence, significantly improving training efficiency.
- **Long-Range Dependencies**: Self-attention facilitates the direct modeling of long-range dependencies without the vanishing gradient problem.
- **Scalability**: The architecture scales well with data and model size, as evidenced by its success in large-scale pre-training and fine-tuning frameworks like BERT [33] and Generative Pre-trained Transformer (GPT) [17].

3.8.3 Applications and Impact

The impact of the Transformer architecture extends beyond Natural Language Processing (NLP). In CV, models like ViTs have demonstrated competitive performance on image classification tasks [37]. Additionally, Transformers have been adapted for various other domains, including speech recognition, reinforcement learning, and even protein structure prediction.

The Transformer architecture represents a paradigm shift in sequence modeling. Its innovative use of self-attention, combined with other architectural components, addresses many of the limitations inherent in previous models. As ongoing research continues to refine and expand its applications, the Transformer model remains a cornerstone of modern ML.

The Transformer architecture, introduced by Vaswani et al. in their seminal paper *Attention is All You Need* [159], has revolutionized various fields of

ML, including NLP, CV, and more. This section explores the fundamental components and principles of the Transformer architecture, its advantages over previous models, and its broad applicability.

3.8.4 Background and Evolution of Transformer Models

Before the advent of Transformers, RNNs and their variants, such as LSTM and GRUs, were predominant in sequence modeling tasks. These models suffered from limitations in capturing long-range dependencies due to their sequential nature and difficulty in parallelization.

The introduction of the Transformer model marked a significant departure from these sequence-based approaches by leveraging the self-attention mechanism, which allows for the direct modeling of relationships between all elements in a sequence.

3.8.5 Core Components of the Transformer

3.8.5.1 Position-wise Feed-Forward Networks

Each position in the sequence is processed independently and identically by a fully connected feed-forward network (FFN), which consists of two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3.26}$$

3.8.5.2 Positional Encoding in Transformers

Since the Transformer lacks a built-in notion of sequence order, positional encodings are added to the input embeddings to provide information about the relative or absolute position of each element in the sequence. These encodings can be learned or fixed and are typically derived from sine and cosine functions of different frequencies as explained in Section 3.8.1.1 by Equations 3.24 and 3.25.

3.8.6 Vision Transformers (ViT)

ViTs have adapted the Transformer architecture for image recognition tasks, demonstrating competitive performance with traditional convolutional neural networks (CNNs). Introduced by Dosovitskiy et al. [37], ViTs divide an image into patches and process them as a sequence of tokens similar to words in a text sequence. Each image patch is linearly embedded, and positional encodings are added before feeding them into the Transformer model.

3.8.7 NLP Large Language Models (LLM)

The Transformer architecture has also been instrumental in developing Large Language Models (LLMs) such as BERT (Bidirectional Encoder Representations from Transformers) [33] and GPT (Generative Pre-trained Transformer) [17]. These models have achieved state-of-the-art results in a variety of NLP tasks by pre-training on vast amounts of text data and fine-tuning on specific tasks. BERT utilizes a bidirectional approach to capture context from both directions, while GPT employs a unidirectional approach to generate coherent and contextually relevant text.

3.8.8 Advantages of the Transformer Architecture

The Transformer architecture offers several advantages over traditional models:

- **Parallelization**: Unlike RNNs, Transformers allow for parallel computation of all elements in a sequence, significantly improving training efficiency.
- **Long-Range Dependencies**: Self-attention facilitates the direct modeling of long-range dependencies without the vanishing gradient problem.
- **Scalability**: The architecture scales well with data and model size, as evidenced by its success in large-scale pre-training and fine-tuning frameworks like BERT [33] and GPT [17].

3.8.9 Applications and Impact

The impact of the Transformer architecture extends beyond NLP. In CV, models like ViTs have demonstrated competitive performance on image classification tasks [37]. Additionally, Transformers have been adapted for various other domains, including speech recognition, reinforcement learning, and even protein structure prediction.

The Transformer architecture represents a paradigm shift in sequence modeling. Its innovative use of self-attention, combined with other architectural components, addresses many of the limitations inherent in previous models. As ongoing research continues to refine and expand its applications, the Transformer model remains a cornerstone of modern ML.

Chapter 4

Computer Vision

CV, often synonymous with Machine Vision (MV) in industrial and automation contexts, is a pivotal field within computer science that focuses on enabling machines to interpret and understand visual information from the world. This capability is fundamental for a myriad of applications, from automated inspection systems in manufacturing to advanced diagnostics in healthcare.

The field of CV encompasses a wide array of technologies and techniques, each tailored to specific types of visual inputs and aimed at particular applications. The primary goal is to automate tasks that require visual recognition and analysis—tasks traditionally performed by human vision. With the rapid advancements in Artificial Intelligence (AI) and ML, CV has seen a significant transformation, enhancing both its capabilities and its range of applications.

4.1 Computer Vision in Industrial Processing

CV plays a crucial role in industrial processing, automating complex tasks such as quality control, production monitoring, and defect detection. These systems dramatically enhance productivity and accuracy across various sectors, including pharmaceuticals, automotive manufacturing, and food processing.

4.1.1 Automated Quality Control

In sectors where quality assurance is paramount, CV systems leverage DL and GAN architectures to perform precise inspections at speeds unattainable by

human operators. For instance, the integration of GAN-based architectures helps detect minute anomalies in high-speed production lines, significantly reducing the incidence of defects. The use of advanced ML techniques enhances the capabilities of these systems to adapt to different production environments, ensuring high-quality outputs consistently [44].

4.1.2 Production Efficiency

CV also enhances production efficiency by monitoring operational workflows in real time. Advanced ML models can predict equipment failures and process deviations, facilitating preemptive maintenance that minimizes downtime and ensures continuous production flow. These intelligent systems are integral in modern manufacturing setups, optimizing operations and reducing waste.

4.2 Blob Analysis: Classical Image Processing Techniques

Computer vision initially relied heavily on classical image processing techniques, which are algorithmic and mathematical operations designed to perform direct manipulations on raw image data. Among these classical methods, blob analysis stands out due to its effectiveness in feature extraction from binary images.

4.2.1 Blob Analysis with Halcon and OpenCV

Blob analysis involves identifying and labeling regions of connected pixels that share similar characteristics. This technique is foundational in tasks such as object localization, shape recognition, and motion analysis. Both Halcon and OpenCV provide extensive support for blob analysis:

- **Halcon**: This software includes sophisticated tools for blob segmentation, which can categorize image regions based on intensity, texture, or motion, and features advanced algorithms for object counting and positioning [155].

- **OpenCV**: Developers utilize functions like `findContours` to detect contours in a binary image, which are further used for shape analysis and object detection [27].

While blob analysis is highly reliable and precise in controlled environments and can be executed quickly due to its deterministic nature, it faces limitations in adaptability and requires extensive parameter tuning to handle varied settings or sample variations. Its inherent rigidity also makes it less suitable for complex or unstructured environments.

Chapter 5

Machine Learning in Computer Vision

Machine Learning (ML) and in particular Deep Learning (DL), has revolutionized the field of Computer Vision (CV) by enabling machines not just to see, but also to understand and interpret visual information with remarkable accuracy, developing models that can perform complex visual tasks. This chapter explores how ML paradigms—supervised, unsupervised, and semi-supervised—have evolved and significantly contributed to advances in CV technologies.

5.1 Applications in Computer Vision Tasks

The integration of ML into CV has transformed numerous industries, from autonomous driving to healthcare diagnostics, by enhancing the ability of systems to analyze and interpret visual data without human intervention. The journey began with simple pattern recognition models in the late 20th century and has progressed to complex DL architectures that can outperform humans in tasks like object recognition and facial identification [91].

Supervised learning, the most prevalent form of ML in CV, leverages large amounts of labeled data to train models that can predict outcomes based on new, unseen data. This approach has led to the development of robust CNNs, which are now fundamental to image and video recognition tasks (Krizhevsky, Sutskever, & Hinton, 2012) [88]. The success of CNNs in the ImageNet chal-

lenge is a testament to their capabilities and has spurred a significant amount of research into network architectures and learning algorithms [139].

Unsupervised learning, on the other hand, deals with identifying patterns in data without the guidance of a known outcome or label. Techniques such as clustering and principal component analysis have opened avenues for feature extraction and data compression, which are crucial in handling the vast amounts of data generated by modern CV systems [158].

Semi-supervised learning, a hybrid approach, utilizes both labeled and unlabeled data to improve learning accuracy. This method is particularly useful in CV applications where obtaining large sets of labeled data is impractical. Techniques developed under this paradigm, such as generative models and self-training algorithms, have significantly improved the efficiency and scalability of ML models in real-world applications [176].

Each of these learning paradigms has uniquely contributed to the field of CV, creating systems that are not only more accurate but also more adaptable to dynamic environments. This chapter will delve into each of these paradigms, exploring their fundamental theories, key developments, and their pivotal roles in advancing the capabilities of CV technologies.

5.1.1 Advances in Machine Learning Techniques for Quality Control and Beyond

The pharmaceutical industry, with its stringent Quality Control (QC) requirements, exemplifies the shift from classical methods to ML. While classical methods like blob analysis have traditionally been used for inline checks due to their reliability and speed, ML algorithms offer a significant enhancement by enabling the automation of QC with greater adaptability and systematicity.

5.1.1.1 Machine Learning in Quality Control

Unsupervised and semi-supervised Machine Learning (ML), especially Deep Learning (DL) and GAN-based architectures, have become integral in modern industrial settings due to their ability to handle complex and variable data. These technologies excel at generalizing across diverse feature distributions and managing unbalanced datasets, where defects are significantly less frequent

than non-defective samples.

The application of gradient-based detection techniques combined with neural networks, as demonstrated by the GRD-Net in Chapters 8 and 9, showcases the ability of these systems to maintain robust feature detection even under variable environmental conditions [34]. This methodology addresses the challenges of the "black box" nature of deep learning, enhancing transparency and reliability which are critical in sectors such as pharmaceutical, healthcare and autonomous vehicle technology.

Explainability of neural network results is particularly crucial in the pharmaceutical field where the qualification of machines and processes by regulatory agencies like the *Food and Drug Administration (FDA)* and *European Medicines Agency (EMA)* is mandatory. Techniques such as Anomaly Segmentation Network (AnoSeg) [170] are often employed to provide foundational explanations for the results produced by ML algorithms. This step is essential for ensuring compliance with stringent regulatory standards, facilitating the adoption of ML technologies in critical areas of pharmaceutical manufacturing [149].

A specific instance of this application is the integration of a GAN-based architecture for anomaly detection on high-speed production lines for pharmaceutical Blow Fill Seal (BFS) vials, it will be discussed in Chapter 8. This demonstrates how ML can not only improve the efficiency and reliability of production processes but also significantly reduce waste and increase safety by early detection of irregularities.

Moreover, advancements in machine learning algorithms have led to their application in other areas of quality control such as predictive maintenance, supply chain optimization, and even environmental monitoring, broadening the scope and impact of ML in industrial applications [150].

5.1.2 Machine Learning for Computer Vision in Other Fields of Applications

ML, particularly DL, has significantly advanced the field of computer vision across various applications beyond quality control. In medical diagnosis, DL models enhance radiological, pathological, and ophthalmological image anal-

ysis, improving accuracy in disease detection and treatment efficacy. Models such as CAT-Unet [34] have advanced medical image segmentation, while deep perceptual autoencoders [147] have improved anomaly detection in medical imaging, facilitating early and more accurate diagnosis. These technologies not only streamline diagnostic processes but also enable early detection of diseases across several specialties, contributing to better patient outcomes and reduced healthcare costs [162, 99, 42, 59].

5.2 Supervised Learning

Supervised learning remains the dominant machine learning (ML) approach in computer vision (CV), tasked with learning from labeled data to predict outcomes for new, unseen data. This paradigm is extensively applied in image classification, object detection, and semantic segmentation, leveraging labeled datasets to train models that can generalize from past experiences to make accurate predictions on new data.

5.2.1 Applications in Image Classification

Image classification, the task of assigning an image to one of several predefined categories, has been a primary driver of ML applications in CV. The advent of deep learning, particularly deep convolutional neural networks (CNNs), has dramatically enhanced the performance of image classification systems. Pioneering networks such as AlexNet, introduced by Krizhevsky et al. [88], have utilized deep layered architectures to learn complex feature hierarchies directly from raw image pixels. This breakthrough was significantly extended by He et al. (2016) with the introduction of ResNet, which employs residual connections to train very deep networks capable of achieving remarkable accuracy on challenging image datasets [64]. The innovative architecture of ResNet helps mitigate the vanishing gradient problem that often occurs in deeper networks, thereby preserving the training efficacy across many layers.

These models have not only achieved state-of-the-art accuracy on benchmark image classification tasks but have also established CNNs as fundamental to many other CV tasks, catalyzing further research and development in the

field.

5.2.2 Object Detection and Localization

Supervised learning extends beyond image classification to include detecting and precisely localizing objects within images. This capability is crucial for applications such as autonomous driving and video surveillance, where understanding the context and location of objects within the visual frame is necessary for making informed decisions. Early breakthroughs in this area were achieved with the development of R-CNN by Girshick et al. (2014) [51], which combined high-capacity CNNs with region proposal techniques to localize and classify objects within an image. This approach was further refined with the introduction of Fast R-CNN and Faster R-CNN by Ren et al. (2015) [133], which enhanced the speed and accuracy of these detections.

Moreover, the introduction of YOLO (You Only Look Once) by Redmon et al. (2016) [131] marked a paradigm shift by treating object detection as a single regression problem, directly predicting bounding boxes and class probabilities in one evaluation. This method significantly accelerates object detection, enabling real-time performance which is vital for applications like real-time surveillance and autonomous navigation.

5.2.2.1 Semantic Segmentation

Semantic segmentation involves classifying each pixel in an image into a predefined category, crucial for detailed image understanding required in medical image analysis and autonomous vehicle navigation. Techniques such as the Fully Convolutional Network (FCN) by Long et al. (2015) [101] and U-Net by Ronneberger et al. (2015) [136] have been fundamental in advancing this area. These models efficiently learn spatial hierarchies through end-to-end training on whole images, enabling precise delineation of object boundaries at the pixel level.

5.2.2.1.1 Semi-Supervised Semantic Segmentation for Anomaly Detection Integrating semi-supervised learning with semantic segmentation has opened new avenues for efficiently utilizing unlabeled data alongside labeled

examples. This hybrid approach is particularly beneficial for anomaly detection in medical and pharmaceutical imaging, where obtaining a large amount of labeled data can be prohibitively expensive and time-consuming. By leveraging unlabeled data, researchers can significantly improve the model's ability to generalize and detect novel anomalies, crucial for effective diagnosis and treatment planning in healthcare settings. Recent methods, such as the Spatial-aware Attention Generative Adversarial Network (SAGAN), highlight the effectiveness of this approach by utilizing attention mechanisms to restore and enhance anomaly-specific features, especially where labeled data is scarce [172].

5.3 Unsupervised and Semi-Supervised Learning

As computer vision (CV) confronts increasingly complex and voluminous datasets, often characterized by imbalance, the role of unsupervised and semi-supervised learning becomes ever more critical. These methods excel without the need for comprehensive labeled data, uncovering latent patterns and structures that are indispensable for tasks like clustering, dimensionality reduction, and feature learning. The high costs and extensive time required for manual labeling make these techniques especially valuable in CV, where data may not only be vast but also partially labeled or completely unlabeled.

Unsupervised learning is crucial for exploring data distributions and identifying unseen patterns, beneficial in scenarios where class labels may not be previously defined. Semi-supervised learning, which combines a minimal amount of labeled data with a larger volume of unlabeled data, enhances model robustness and accuracy in environments where acquiring complete labels is unfeasible. This approach is particularly effective in handling class imbalances and label known a priori, common in fields such as autonomous driving, medical and pharmaceutical imaging, where it can significantly improve the generalization capabilities of CV systems.

5.3.1 Out-of-Distribution Detection

Detecting OOD samples is essential for the reliability of CV systems, particularly in safety-critical fields such as QC in pharmaceutical and medical imaging, autonomous driving and medical diagnostics. OOD detection techniques assess whether incoming data deviate from the training distribution, helping to prevent misclassifications or errors.

Traditional OOD detection methods use threshold-based evaluations of features or predictive uncertainties. However, more sophisticated approaches, such as those by Hendrycks and Gimpel (2017) [69], employ neural networks to learn features of in-distribution data, thereby improving detection sensitivity and specificity. Additionally, Liang et al. (2018) [95] enhanced detection capabilities by adjusting the softmax temperature in neural networks, which helps distinguish between in-distribution and OOD samples more effectively.

The implementation of advanced OOD detection methods is critical to ensure the robustness and reliability of ML systems. By identifying nonconforming data, these methods safeguard against errors that could result in operational failures, such as in pharmaceutical vials QC, autonomous vehicle navigation or patient diagnosis. As CV technologies advance, refining OOD detection algorithms will continue to be a key focus.

5.3.2 Reconstruction Models

Reconstruction models such as AEs and GANs are fundamental to modern machine learning, particularly in learning intricate data distributions and generating novel data instances. AEs compress data into a lower-dimensional space through an encoding process and then reconstruct it to its original form in the decoding phase. This dual process aids not only in data compression but also in anomaly detection, where deviations from the reconstructed output signify anomalies. VAEs, a specific type of AEs, introduced by Kingma and Welling (2013) [85], are particularly effective in modeling complex distributions and generating new data that conforms to the learned distributions, offering profound insights into data's inherent structures. Further enhancements in AEs, such as Masked autoencoder (Masked-AE) [62], and VQ-VAE [1, 157], enhance the flexibility and efficiency of AEs in handling diverse data

characteristics, as previously mentioned in 3.3.3 and 3.3.4.

On the other hand, GANs employ a game-theoretic strategy where two models, a generator and a discriminator, compete. The generator aims to produce data indistinguishable from actual data, while the discriminator strives to differentiate between the generated and real data. This dynamic improves the quality of generation, making GANs exceptional tools for creating realistic images and other media forms. Advanced GANs variants like GANomaly [2] and Skip-GANomaly [3], further refine this process, especially in anomaly detection scenarios, by enhancing the generator's ability to reproduce typical data distributions and highlight anomalies effectively. autonomous vehicle training.

5.3.2.1 Diffusion Models and DiffusionAD

As outlined in Section 3.5, diffusion models are a promising class of generative models capable of producing high-quality, realistic images. They operate by converting data to Gaussian noise through a diffusion process and then learning to reverse this transformation via iterative reverse diffusion steps. Sohl-Dickstein et al. (2015) [151] provide a thorough theoretical foundation, explaining the refinement of samples through a Markov chain process.

Building on this foundation, DiffusionAD [5], or Anomaly Detection with Conditioned Denoising Diffusion Models, leverages the principles of diffusion models specifically for anomaly detection. By conditioning the diffusion process on anomalous data, DiffusionAD enables the model to learn the typical data distributions and identify deviations with high precision. This approach has been particularly effective in tasks requiring detailed and precise anomaly detection, such as in medical imaging and quality control in manufacturing sectors. By focusing on the subtle differences between normal and anomalous data, DiffusionAD can detect and localize anomalies with greater accuracy compared to traditional methods.

These sophisticated reconstruction models are pivotal in fields where accurate simulation of real-world data is critical, enhancing applications from medical imaging to autonomous vehicle training.

5.3.3 Advanced Reconstructive and Discriminative Approaches for Anomaly Detection

Advancements in anomaly detection have led to the development of sophisticated models that combine discriminative and reconstructive processes, enhancing Anomaly Segmentation Network (AnoSeg) capabilities in CV. Among these, Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection (DRÆM) and Generative Reconstructive Discriminative Network (GRD-Net) represent cutting-edge approaches tailored to address complex detection tasks with high accuracy.

Developed by Zavrtanik et al. [170], DRÆM introduces a novel anomaly detection methodology through a discriminatively trained model that enhances the embedding space, significantly increasing sensitivity to subtle surface anomalies. This technique is adept at detecting minute texture and pattern discrepancies that are typically overlooked by standard models, offering a refined tool for identifying surface anomalies with heightened accuracy.

Building upon the principles established by DRÆM [170], GRD-Net, further advances these concepts by incorporating residual structures and an attention mechanism. These enhancements enable GRD-Net to focus specifically on regions of interest within an image, which is crucial for processing scenes with subtle or sparse anomalies. The attention module in GRD-Net dynamically adjusts focus, directing computational resources to areas most likely harboring anomalies, thus improving the overall detection accuracy and efficiency.

Both DRÆM and GRD-Net utilize a blend of generative, reconstructive, and discriminative techniques to effectively tackle anomaly detection challenges. DRÆM focuses on discriminatively enhancing the embedding space, while GRD-Net extends this approach by integrating attention mechanisms that adaptively focus on anomaly-prone regions. This synergy between discriminative training and focused attention mechanisms marks a significant advancement in the field, pushing the boundaries of what automated systems can achieve in terms of sensitivity and specificity in anomaly detection tasks.

5.3.4 Embedding Similarity Models

In many CV applications, the ability to measure the similarity between different data representations is crucial. Techniques in this domain are vital for a range of tasks, from facial recognition to image retrieval, where the primary objective is to quantify the similarity between images based on their embedded features. These techniques rely heavily on distance metrics or similarity scores to effectively compare feature embeddings, such as those implemented in the FaceNet system by Schroff et al. (2015) [143].

5.3.4.1 Patch based

Patch-based methods like PatchCore [137] and PaDiM [32] provide robust solutions for anomaly detection by comparing image patches, making them highly suitable for industrial quality control. These methods focus on localized regions within an image to detect anomalies, crucial for identifying minute defects that impact product quality. These models have shown high efficacy in environments where precise anomaly localization is necessary, as demonstrated by Bergman et al. (2020) [12].

5.3.4.2 Advanced Techniques for Flow Modeling

Flow-based models such as FastFlow [122] and CFlowAD [58] employ normalizing flows to build flexible and powerful generative models that are particularly useful in unsupervised anomaly detection. These models are adept at capturing the complex distributions typical in high-dimensional data, facilitating the modeling and generation of realistic data distributions. Kingma and Dhariwal (2018) [83] highlighted their potential in producing high-quality samples that effectively mirror the training distribution.

5.3.4.3 Innovative Anomaly Detection Approaches

Newer strategies such as the GLASS [98], along with EfficientAD [153], incorporate advanced learning techniques that synthesize anomalies for better model training. These methods leverage gradient ascent to generate realistic anomalies, improving the model's ability to detect and localize defects across various industrial scenarios.

5.3.5 Knowledge Distillation

Knowledge distillation is a technique where a smaller, more computationally efficient model, often referred to as the student, is trained to emulate the behavior of a much larger and typically more powerful model, known as the teacher. This approach is invaluable in deploying powerful machine learning (ML) models to mobile devices and embedded systems, where computational resources are limited. Hinton et al. (2015) [71] originally popularized this concept, demonstrating how the knowledge from a cumbersome model can be transferred to a smaller model that is more suitable for deployment in resource-constrained environments.

Beyond its initial applications, knowledge distillation has been extensively explored for improving the performance of models in various domains, including anomaly detection. Techniques such as EfficientAD [9] and "Uninformed Students: Student–Teacher Anomaly Detection with Discriminative Latent Embeddings" [15] utilize knowledge distillation to enhance the detection capabilities of lightweight models, allowing them to perform complex anomaly detection tasks typically reserved for much larger models. This is particularly useful in scenarios like real-time processing on edge devices where both accuracy and efficiency are crucial.

Additionally, recent advancements have explored the use of knowledge distillation in specialized areas such as semi-supervised learning and unsupervised anomaly detection. For instance, the work by Lopez-Paz et al. (2016) [103] introduces the concept of distilling knowledge in teacher-student setups where the student learns under the guidance of the teacher model without direct access to large labeled datasets.

The development of these techniques has broadened the scope of knowledge distillation, making it a cornerstone strategy not just for model compression but also for enhancing model robustness and adaptability in diverse ML applications.

Part III

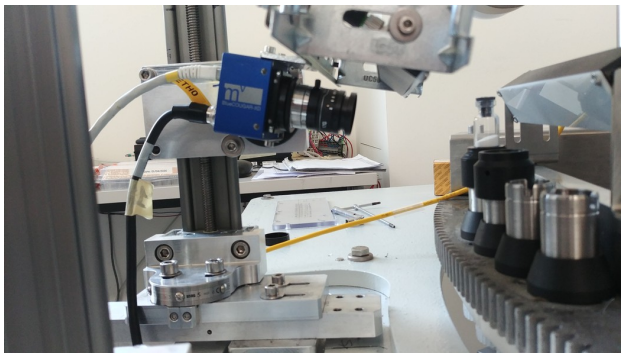
Anomaly Detection in Industrial Processes for Quality Control

Chapter 6

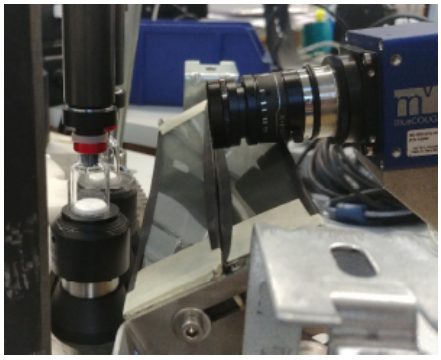
Introduction to Anomaly Detection in Industrial Processes

Anomaly Detection (AD) in Computer Vision (CV) is a task increasingly used in the industrial sector. The vast majority of Quality Control (QC) processes in the pharmaceutical sector still rely on traditional algorithmic CV techniques, such as blob analysis. This is primarily because major drug regulatory agencies face challenges in validating Machine Learning (ML) algorithms due to their low explainability. However, the increasing complexity of packaging and the extension of these controls to more complex products have made these traditional algorithms increasingly complex, cumbersome, and difficult to structure.

Another crucial point is that an algorithmic approach is typically "tailor-made" for a given set of images, which is usually quite limited, because every modification to the algorithm must be manually verified by the developer. Clearly, such an approach lacks the ability to generalize due to the variability of the product, as well as the limited capacity of image processing operations to describe the features extracted by the algorithm. Nowadays, with the ability to acquire vast sets of images, as in Figure 6.1, ML algorithms are gaining increasing importance due to their capability to process many more examples and extract salient features, thereby better generalizing the overall concept of the product being tested.



(a) Carusel station



(b) Test bench station

Figure 6.1: Acquisition stations.

6.1 Main Real-Case Issues

In real-world industrial scenarios, particularly in production lines, several challenges are encountered that are often not thoroughly addressed in the literature. This is primarily due to either the lack of large, open, and truly realistic datasets or the predominant focus on prototyping and mathematical modeling. This focus can sometimes result in a disconnect between the proposed solutions and their feasibility in an industrial context.

The main real-case issues can be summarized as follows:

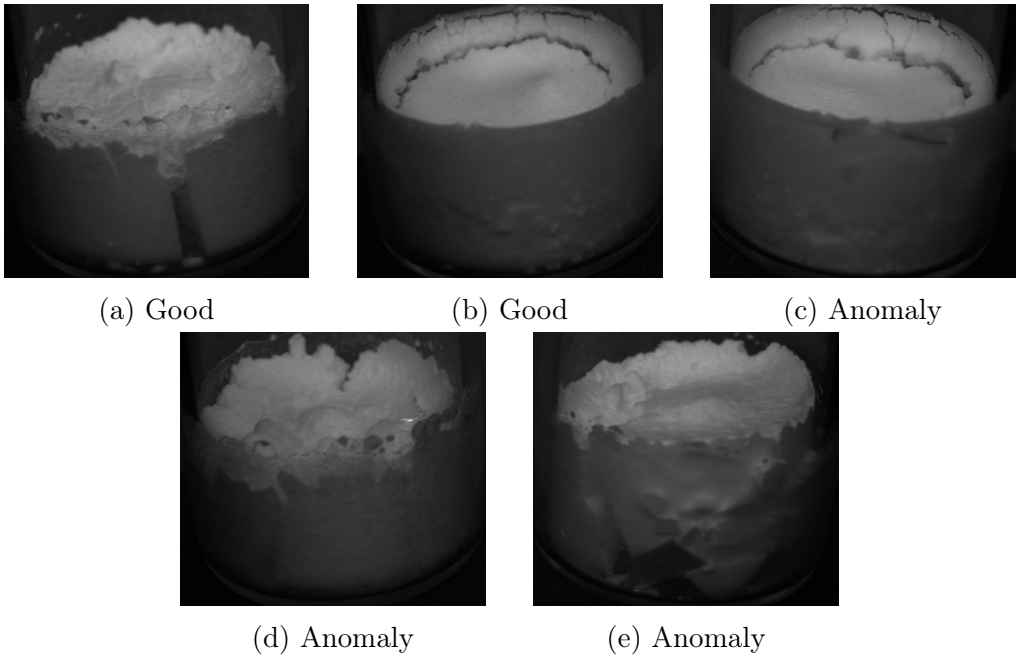
- Vast and highly complex datasets, often with significant intrinsic variability in the product or its content.
- Highly unbalanced datasets.
- Inference time as a critical parameter in system design.
- Cost-performance ratio and hardware limitations, especially during inference.

6.1.1 Dataset Complexity

Most industrial datasets are vast and extremely complex due to the variability of the product, and particularly in the cases that will be discussed in the pharmaceutical field, the variability of the content. This complexity makes training the network particularly challenging, as it must generalize a rule and

understand the salient features that distinguish the various classes in a supervised approach or differentiate between regular and anomalous products in a semi-supervised approach.

The MVTec datasets for AD [14] provide a valuable benchmark for testing the performance of new architectures. However, they are often quite limited and not very realistic. In fact, most new architectures now achieve performance that approaches 100% accuracy on images and 100% AUROC for pixel-wise detection.



(a) Good (b) Good (c) Anomaly

(d) Anomaly (e) Anomaly

Figure 6.2: Products (a) and (b) are regular, (c) has a fiber stuck in the cake, (d) a piece of glass floating on the top of the cake, (e) a rubber stuck on the bottom of the cake.

Figure 6.2 shows two good products and three anomalies with a freeze-dried cake inside the glass vial. The vials themselves exhibit very little variability, thanks to the tubular glass formation process, but the freeze-dried cake is highly variable and often breaks during the formation process, making it challenging to distinguish a good product from a rejection.

Other example of liquid filled BFS plastic vials[1] could be found in "GRD-Net: Generative-Reconstructive-Discriminative Anomaly Detection with Re-

[1]Because of a Non Disclosure Agreement (NDA), the *flag* (the upper part) was blurred to conceal the company logo.

gion of Interest Attention Module" [43] and "Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line" [44]. An example is reported in Figure 6.3.

(a) Good

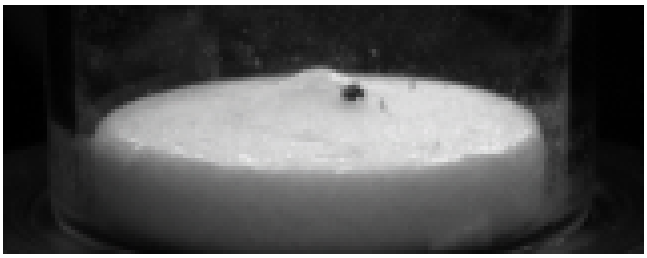
Figure 6.3: Liquid filled product[1] [44].

6.1.2 Dataset Imbalance

Another significant challenge encountered in real-world industrial scenarios is the imbalance within datasets. In many cases, the majority of the data

belongs to a single class, while the minority class, which is often of greater interest—such as defective products in quality control applications—is underrepresented. This imbalance can lead to biased models that perform well on the majority class but fail to accurately detect or classify instances from the minority class. Addressing this issue is crucial for developing robust machine learning models that are effective in practical applications [77, 61, 19].

It is relatively easy to collect and acquire a vast number of examples of production-quality goods. However, creating a comprehensive library of defects that covers the full range of possible issues is both expensive and challenging. This difficulty arises because the collection of defects typically requires an extended period of observation during production and must be handled manually. Moreover, generating defective products ad-hoc is not only costly and complex but also likely to introduce bias. This bias often stems from the limited experience of manual operators, particularly when dealing with new products, which can skew the representation of defects. The defect shown in Figure 6.4 was produced in the laboratory. It features a perfectly formed cake, which is not realistic compared to those typically observed in production.



(a) Non-realistic defect

Figure 6.4: A defect produced in the laboratory with a non-realistic, perfectly formed cake.

6.1.3 Inference Time

A vast majority of academic papers focus on outperforming the state-of-the-art models in terms of accuracy and other performance metrics. While these advancements are crucial for pushing the boundaries of technological innovation, they often overlook practical considerations such as inference time. The speed at which a model can make predictions—known as inference time—is partic-

ularly important in industrial applications where real-time or near-real-time decision-making is required.

In many cases, models that achieve superior accuracy do so at the cost of increased computational complexity, leading to longer inference times. This trade-off can render such models impractical for deployment in production environments, where latency and throughput are critical factors. Despite its importance, inference time is rarely emphasized in the literature, with only a few notable exceptions [9, 44] that address the feasibility and time efficiency of models in real-world applications.

In industrial settings, where large volumes of data need to be processed quickly, the balance between model accuracy and inference time becomes a key consideration. Models that are not optimized for inference time may introduce delays in the production process, potentially leading to inefficiencies and increased operational costs. Therefore, it is essential to develop and evaluate models not only based on their predictive performance but also on their ability to deliver timely results in a production environment.

6.1.4 Cost-Performance Ratio and Hardware Limitations

Another crucial aspect in the design and development of industrial machine learning (ML) architectures is the cost-performance ratio. This ratio is a critical factor that directly influences the choice of hardware and, consequently, imposes limitations on the complexity and scalability of the models that can be applied.

In industrial settings, the deployment of ML models often requires a careful balance between performance and cost. High-performing models typically demand more computational resources, which can significantly increase the cost of the necessary hardware [31]. For instance, models that require powerful GPUs or specialized accelerators may be prohibitively expensive to deploy at scale, particularly in environments with budget constraints or where cost-efficiency is a priority [79].

These hardware limitations can restrict the type and complexity of models that can be used in practice. While more complex models may offer better predictive performance, they may also require hardware that is either too costly or unavailable in certain industrial contexts [70]. As a result, there is often

a need to compromise, selecting models that provide an acceptable trade-off between performance and hardware requirements.

Moreover, the scalability of ML solutions is often constrained by these hardware limitations. As the size of the data or the number of tasks increases, the demand on hardware resources also grows [8]. Ensuring that the deployed models can scale effectively without incurring prohibitive costs is a key challenge in the industrial application of ML.

Therefore, when designing ML architectures for industrial use, it is essential to consider not only the accuracy and efficiency of the models but also the cost-performance ratio. This consideration ensures that the solutions are not only technically feasible but also economically viable, making them more likely to be adopted in real-world applications.

Among the most cited architectures in the literature are embedding similarity-based systems, which often rely on a *Memory Bank* [137, 32, 165, 63, 54, 113]. This *Memory Bank* collects embeddings extracted from a pre-trained or knowledge-distilled network using a large and sparse dataset, such as ImageNet [139] and its derivatives. In essence, the *Memory Bank* stores a collection of feature embeddings, extracted from one or more deep layers of the network, that represent common patterns within the dataset, which can be referenced to detect deviations indicating anomalies. While this mechanism can perform well on limited and relatively simple datasets, such as those provided by MVTec, it faces significant challenges when applied to more complex products. Specifically, more complex products require much larger datasets, and the *Memory Bank* often cannot fit into the on-board cluster RAM due to its size requirements. To reduce the memory load, methods such as greedy coreset reduction or KMeans-based centroid aggregation are sometimes employed. However, these techniques can lead to loss of significant information if embeddings are under-represented, or may still require substantial memory resources for larger datasets, limiting their scalability. Additionally, managing and querying large memory banks to maintain retrieval efficiency can be computationally taxing and may increase inference time. This limitation renders such models impractical for real-world applications. On the other hand, reconstruction-based architectures, such as AEs and GANs, typically incur a higher computational cost due to the increased complexity of their larger models.

6.2 Implemented Approaches

Throughout this study, various cases with distinct challenges and peculiarities were examined. For each case, different approaches were studied and implemented, all aimed at developing a real-time, in-line system.

6.2.1 Feasibility Study of Freeze-Dried Products in Tubular Glass Vials

The first case involves a feasibility study focused on the freeze-dried cake within a tubular glass vial. For this study, the client provided an extensive library of both good products and anomalies. Since the dataset was not heavily imbalanced, the proposed solution was to use a supervised approach based on a pre-trained network. This method is a well-established choice for AD, particularly when the dataset includes a comprehensive range of defects and the distribution between training and testing sets is relatively balanced. Another strength of this approach is its practicality in implementation, as the output directly provides the class to which the input image belongs. Additionally, these networks generally have a lower computational cost compared to those used in reconstruction-based approaches, which will be discussed in the following sections. However, the main drawback is the explainability of the result: it can be challenging to account for which features most significantly contributed to a given classification. Some studies, such as GradCAM [144], attempt to address this by using a heatmap to highlight the most salient features, or areas of the image, that led to the classification. If the defect library is very extensive and diverse, another possible supervised approach could be segmentation, such as semantic segmentation [102]. As will be demonstrated later, this approach yielded excellent results within the dataset provided by the client.

6.2.2 Evaluation of Out-of-Distribution Anomaly Detection in BFS Vial Strips

The second and third cases analyzed in this study both involve strips of vials made from BFS, a type of plastic commonly used in the pharmaceutical industry. This material, increasingly favored due to its lower cost compared to

tubular glass, presents significantly more challenging issues than its predecessor. These challenges stem from the material's irregularity, lower transparency, and a higher likelihood of small, subtle defects, such as scratches, which are more difficult to detect.

6.2.2.1 First Industrial Application: Anomaly Detection and Segmentation in BFS Strips

The first OOD-AD model examined was applied in a real-world industrial process for quality control of pharmaceutical BFS strips of vials. The tests were conducted using an automatic machine from Bonfiglioli Engineering, featuring a rotary carousel equipped with sensors for image acquisition. The training set comprised 230,355 images of vials, captured in three different areas by an online camera during the production process. Due to NDAs, we are unable to display full product images and instead focus on a limited area that covers one of the most critical regions of interest.

Each strip consists of five BFS plastic vials, joined together along the long side and filled with liquid. One of the most challenging areas to analyze is the meniscus region, due to its inherent randomness and variability. The shape of the meniscus, coupled with the potential presence of bubbles underneath or liquid droplets on top, makes it difficult to adequately process this region using traditional blob analysis algorithms.

In this study, we utilized the architecture described in "GRD-Net: Generative-Reconstructive-Discriminative Anomaly Detection with Region of Interest Attention Module" by Ferrari et al. [43]. For each training image, a ROI was prepared, using an assisting tool developed in Bonfiglioli Engineering using Halcon framework, to focus the network's attention on the product. The final output is a heatmap and an anomaly score generated by the second U-Net, as will be further detailed in subsequent sections.

6.2.2.2 Second Industrial Application: Enhancing Anomaly Detection in Lower-Quality BFS Strips

The second OOD-AD model focuses on a product similar to the first but with additional challenges arising from the production process of the vial strips, which results in lower quality. The dataset provided for both training and

testing is much larger, containing approximately 1,000 good products. These products were acquired multiple times after being repositioned by the machine, thus presenting different conditions in terms of position, liquid level, and the presence of bubbles or droplets on the internal surface and the liquid meniscus. This approach effectively leverages data augmentation, providing more diverse data during the training phase.

The architecture used in this study, described in "Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line" by Ferrari et al. [44], called in the context of this Thesis ResGANAD, is inspired by the model applied in the first study (Section 6.2.2.1). However, it has been simplified to meet the strict time requirements and enhanced by improving the reconstruction loss, the residual AE, and the denoising process to produce images more closely resembling the regular ones. In this case, the heatmap is generated by calculating the absolute difference between the original and reconstructed images, as proposed in the GANomaly paper [2]. The anomaly score is computed using the Structural Similarity Index (SSIM) loss, as will be further detailed in subsequent sections.

In the context of this work, the training set is composed of 2,815,200 images obtained from 782 different strips, each containing five vials. The strips were acquired 10 times, with 16 frames and 2 ranked images per acquisition. Each image was divided into 20 patches: 5 regions, one per vial, each subdivided into 4 sub-regions.

A key focus of the implementation is the throughput of the inference, which is integrated into the on-board system and developed using TensorFlow 2.13.0 C++ APIs.

6.2.3 Exploring Embedding Similarity-Based Methods for Large-Scale Applications

This ongoing study aims to make embedding similarity-based technologies feasible, particularly those like PaDiM [32], PatchCore [137], and more recent methods such as PNI [6], which are highly promising but face the intrinsic challenge of *Memory Bank* growth as the number of provided examples increases. Despite technical proposals for dimensionality reduction, such as greedy core-

set subsampling or Principal Component Analysis (PCA), this issue persists.

To address this challenge, a probabilistic approach is proposed that employs an online version of the Mahalanobis distance between the statistical parameters of training embeddings and the deep features extracted from the input. Additionally, a second method is introduced using online clustering to approximate the *Memory Bank* by using centroids as centers of gravity and calculating k-Nearest Neighbours (k-NN) with Euclidean distances.

Chapter 7

Labeled Anomaly Detection with Supervised Learning

The purpose of this chapter is to explore the application of Convolutional Neural Networks (CNNs) for the classification of lyophilized cake vials in real-case scenarios. The research compares various frameworks and technologies, ranging from classical algorithmic methods implemented with the Halcon 18.11 framework to advanced models using pre-trained CNNs with TensorFlow 2.15 backend. This study focuses on products from multiple clients, including those characterized by well-formed cakes and clean glass, as well as those with highly variable cake shapes and occasionally unclean glass.

As discussed in section 5.2, supervised learning remains the dominant Machine Learning (ML) paradigm in Computer Vision (CV). It involves learning from labeled data to predict outcomes for new, unseen data. This approach is extensively utilized in image classification, object detection, and semantic segmentation, leveraging labeled datasets to train models that can generalize from learned experiences to make accurate predictions on new data.

7.1 Overview of the Methods

The following sections will describe the deep learning methodologies employed in this study to tackle the challenges of defect classification in lyophilized cake vials:

- A DL approach using pre-trained CNNs for glass cosmetic defect classification, leveraging sophisticated image processing techniques to enhance the visibility of subtle irregularities.
- A DL approach using pre-trained CNNs for the identification and classification of surface particles, integrating advanced algorithms for accurate particle detection and categorization.
- A blob analysis technique, employing traditional image processing methods to identify and classify surface particulate matter, which complements the deep learning approaches by providing a robust fallback method.

7.2 Image Classification

The products and their corresponding acquired images were categorized into three groups: *good*, *glass defects*, and *particle and cake defects*, as shown in Table 7.1.

Table 7.1: Distribution of Samples Across Different Defect Categories

Category	**Good**	**Glass Defects**	**Particle and Cake Defects**
# of Samples	100	27	36

Among the defects, particle and cake defects are further subdivided into 12 particle samples and 24 defective cake samples. The total count of defects is 63, compared to 100 good products. An example of particle defects is shown in Figure 7.1.

7.2.1 Problem Description

The objective of this study is to identify and classify two primary groups of defects—cosmetic and particulate—within a set of samples. These samples consist of tubular glass vials filled with a freeze-dried cake, prepared in a laboratory setting and provided by the client. Although these products are in significantly better condition than those typically produced on a manufacturing line, they still present a number of complex challenges. The tubular glass used is extremely pure, typically devoid of irregularities unless flawed. However,

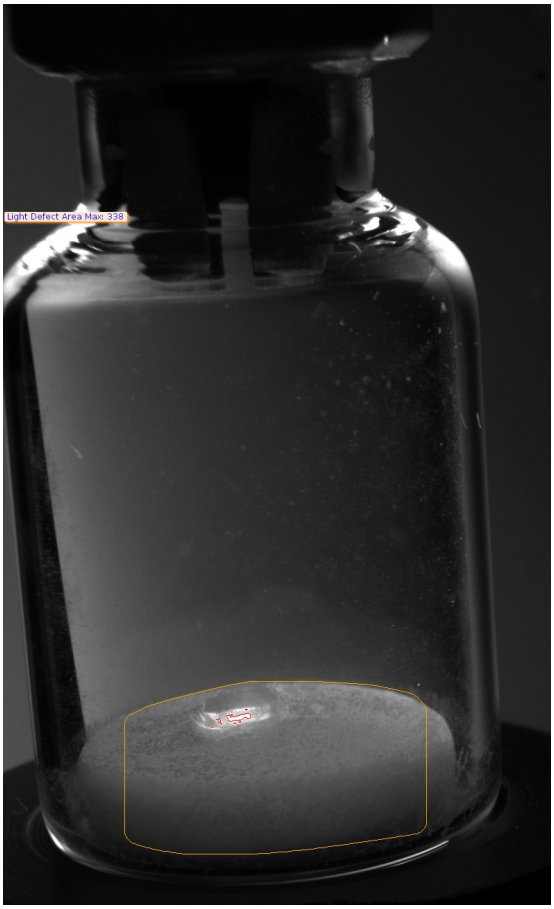


Figure 7.1: An inspected product illustrating particle defects.

the lyophilized product inside does not possess a regular structure and often releases particles that adhere to the inner surfaces of the glass. This internal contamination with dust complicates the defect identification process, as the algorithm must discern these from inherent irregularities in both the glass and the lyophilized product.

To ensure the results are applicable to real-world settings, the study was also conducted on a dataset collected directly from a production line, provided by another client for a different project. This approach allows the research to address the variability in sample quality that occurs in typical production environments.

Furthermore, the study aimed not only to identify whether a product was defective but also to classify it within a list provided by the client during the initial requirements definition. There was no explicit demand from the

client to identify the exact location of the defects, which places a focus on the classification accuracy of the detection algorithm rather than on detailed spatial analysis.

By evaluating both laboratory-prepared and production-line samples, this research seeks to develop and validate an algorithm that can robustly classify defects in a diverse set of conditions. This dual dataset approach enhances the understanding of the algorithm's performance and its potential for implementation in automated quality control systems within the pharmaceutical industry, ultimately aiming to reduce waste and improve product quality.

7.2.2 Classical Approach with CNN

For this part of the study, we opted for a labeled approach using pre-trained CNNs. This decision was driven by the specific goal of classifying products without performing AnoSeg, and because the limited number of samples provided was insufficient to effectively train any kind of reconstructive architecture from scratch. Utilizing pre-trained CNNs offers a robust framework for leveraging existing neural network architectures, which have been extensively trained on large datasets [55]. This approach allows for the adaptation of these models to our specific dataset with minimal need for retraining, thus overcoming the challenge of limited sample size while still achieving reliable classification results [64].

Regarding cosmetic defects, a ResNet18 [64] model was employed because it is the lightest among the residual models pre-trained on ImageNet, which suited the subtle visibility of cosmetic defects in the provided dataset, as depicted in Figure 7.2. In contrast, particle defects, which typically stood out with high contrast against the cake (as in Figure 7.4), were analyzed using MobileNetV3 [76] and a blob analysis algorithm using Halcon (Figure 7.1). The ResNet50v2 [65] was specifically chosen for a separate dataset obtained from production products, where the cake formations were less consistent (Figure 7.3), requiring a deeper and wider network for effective detection.

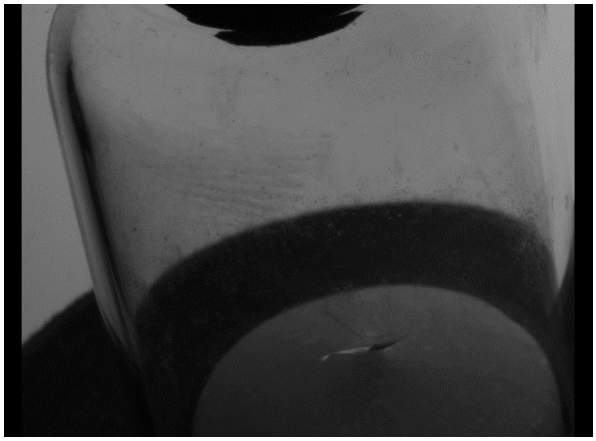


Figure 7.2: The cosmetic defect is a tiny crack in the lower and darker part of the picture

7.2.3 Deep and Wide Residual Networks

Deep and wide residual networks expand upon the traditional DL architectures by incorporating residual connections, which help to combat the vanishing gradient problem that often occurs with very deep networks [64]. These networks are characterized by their depth—number of layers—and their width—number of neurons per layer.

Residual networks, or ResNets, make use of skip connections, or shortcuts to jump over some layers. Typical ResNets, such as ResNet18 or ResNet50, involve layers where the input to a layer is added to its output, helping to preserve the identity of the input throughout the network and thus alleviating the issue of vanishing gradients as the network depth increases [64].

Wide Residual Networks (WRNs) are a variation that adds more neurons to each layer (as shown in Figure 7.5), increasing the network's width, and are found to be more effective when the network needs to learn more complex features or when more training data are available [169]. By increasing the width, WRNs can achieve higher accuracy than their narrower counterparts, often with fewer layers, reducing the computational burden while still capitalizing on the depth provided by residual learning.

This innovative architecture has been broadly adopted in various domains, demonstrating significant improvements over standard CNN models in both academic and industrial applications [64, 169].

Figure 7.3: The particle defect in a real-case production freeze dried cake.

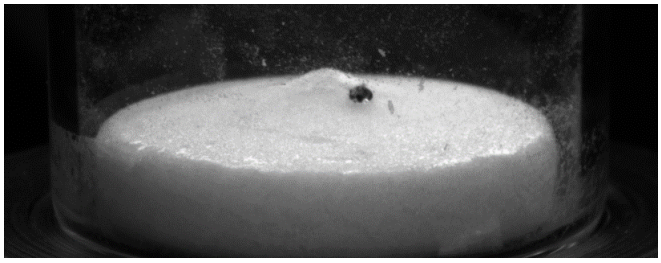


Figure 7.4: The particle defect in a laboratory made freeze dried cake.

7.2.4 Approach Limitations

This approach has several limitations. Primarily, the explainability of the classification results can be difficult to extract and may not be as precise as segmentation-based methods. While this level of detail was not required by the customer in the described case, it is often a critical requirement in fields such as pharmaceuticals. In these cases, Anomaly Segmentation Network (AnoSeg) can be more important than merely classifying defects.

To enhance interpretability, techniques like Gradient-weighted Class Activation Mapping (Grad-CAM) [144] are commonly employed. Grad-CAM provides a visual explanation of the classification decisions by highlighting important regions in the input image. For a detailed mathematical explanation of Grad-CAM and its implementation, please refer to Appendix A.

Furthermore, another significant limitation of this technology lies in its

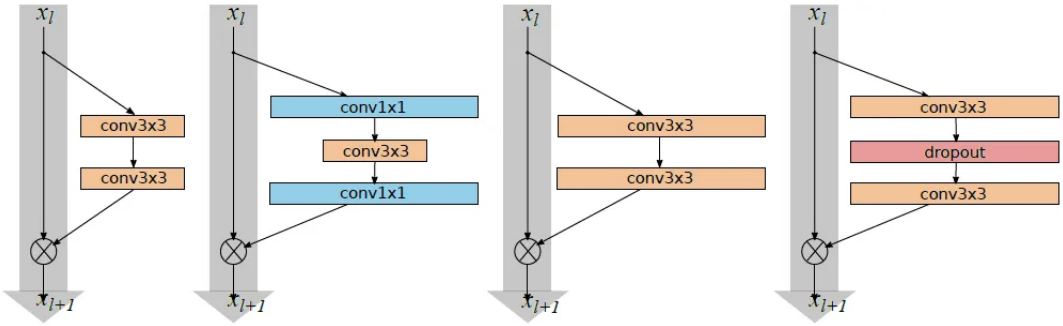


Figure 7.5: Comparison of standard residual block, bottleneck residual block, wide residual block, and dropout residual block from Zagoruyko et al. [169], illustrating the increased width in WRNs.

future-proofing, particularly in the event of production drifts or the appearance of novel, previously unseen defects. In both scenarios, the network may misclassify defects. This issue is especially prevalent in production lines, where the defect library provided is based on prior experience and defects defined by acceptance criteria. However, when new materials are introduced, such as BFS, where experience is more limited and the range of potential defects is much broader compared to tubular or even molded glass, the likelihood of encountering something previously unconsidered increases significantly. In such cases, the network may lack a corresponding class for the defect and is likely to misclassify it, with the risk of assigning it to the class of acceptable (good) products.

7.2.5 Related Work

The application of CNNs in image classification has been extensively explored, leading to significant advancements in model architecture and interpretability. LeCun et al. [92] laid the foundation for modern deep learning approaches with their introduction of CNNs. Over time, these architectures have evolved, becoming deeper and more complex to address increasingly challenging tasks.

A major advancement in CNN architecture was the introduction of ResNets by He et al. [64, 65], which addressed the vanishing gradient problem through residual connections. This innovation allowed for the successful training of much deeper networks, such as ResNet50, which has become a benchmark in

image classification.

Building on the success of ResNets, Zagoruyko and Komodakis [169] proposed WRNs, demonstrating that increasing the network's width can improve performance with fewer layers, thereby reducing computational demands.

In addition to architectural improvements, the interpretability of deep learning models has become increasingly critical, particularly in sensitive domains like healthcare and pharmaceuticals. Grad-CAM [144] is a widely adopted technique that provides visual explanations for CNN decisions by highlighting relevant regions in input images, thereby enhancing model transparency [141].

Despite these advancements, challenges remain in real-world applications, particularly in handling production drifts or new, unseen defects. This is especially critical in industries like pharmaceuticals, where defect libraries are based on historical data and acceptance criteria. For instance, with the introduction of new materials such as BFS technology, the range of potential defects expands, increasing the risk of misclassification [38].

To address these challenges, techniques like transfer learning [121] and domain adaptation [50] have been proposed, enabling models to adapt to new data with knowledge from related tasks or domains. However, significant distribution shifts between source and target domains remain a limitation [127].

In summary, while CNNs and related architectures have seen substantial development, challenges related to interpretability and robustness persist, especially in industrial applications. Ongoing research focuses on improving the generalization of these models to new and unseen defects while maintaining transparency in their decision-making processes.

7.3 Semantic Segmentation

Semantic segmentation is a critical task in CV where the goal is to classify each pixel in an image into a predefined category. Unlike image classification, which assigns a single label to an entire image, semantic segmentation provides a much finer level of detail by localizing and classifying objects within the image. This capability is particularly advantageous in applications where understanding the precise location and boundaries of objects is essential, such

as in pharmaceutical and medical imaging, autonomous driving, and quality control in manufacturing.

7.3.1 Classical Approach with U-Net

One of the most prominent architectures for semantic segmentation is the U-Net [136], which was originally developed for biomedical image segmentation. The U-Net architecture is characterized by its symmetric encoder-decoder structure, where the encoder gradually reduces the spatial dimensions of the input while increasing the depth of the feature maps, and the decoder gradually reconstructs the spatial dimensions while reducing the depth. Skip connections between corresponding layers in the encoder and decoder allow for the preservation of spatial information, which is crucial for accurately delineating object boundaries.

The U-Net has proven to be highly effective in various segmentation tasks due to its ability to capture both contextual information and fine-grained details. In our study, however, the application of U-Net was constrained by the limited amount of data provided by the customer. Semantic segmentation models like U-Net typically require large, well-annotated datasets to achieve high accuracy, as the model needs to learn the intricate patterns associated with different object classes at the pixel level.

7.3.2 Approach Limitations

Despite the powerful capabilities of semantic segmentation, there are several limitations associated with its application. One major limitation is the requirement for extensive labeled data. In our study, the dataset provided by the customer was not sufficient to fully leverage the potential of semantic segmentation. This limitation is particularly challenging because pixel-level annotation is both time-consuming and labor-intensive, often requiring domain expertise.

Moreover, semantic segmentation suffers from similar issues as those encountered in image classification, especially in terms of generalization to new and unseen data. Since semantic segmentation is a supervised learning task, the model's performance heavily depends on the quality and diversity of the

labeled training data. If the training data does not adequately represent the variability present in real-world scenarios, the model may fail to accurately segment novel or unexpected objects, leading to misclassifications.

Another limitation is related to the interpretability of the model's decisions. While semantic segmentation provides pixel-level localization of objects, which inherently improves explainability by showing exactly where an object is located within the image, it still struggles with the same issues as classification models in terms of understanding why a particular segmentation decision was made. Techniques like Grad-CAM have been adapted for segmentation tasks to provide visual explanations [144], but these are still relatively coarse and may not always provide the level of detail required for critical applications.

7.3.3 Related Work

Semantic segmentation has been extensively studied in the literature, with various architectures being proposed to improve accuracy and efficiency. The U-Net architecture, as mentioned earlier, has become a standard benchmark in the field, particularly in medical image analysis [136]. Other notable architectures include DeepLab [21], which introduced atrous convolution and Conditional Random Fields (CRFs) to refine segment boundaries, and PSPNet [173], which utilized a pyramid pooling module to capture contextual information at multiple scales.

In addition to architectural advancements, there has been significant research on improving the robustness and generalization of semantic segmentation models. For instance, transfer learning and domain adaptation techniques have been explored to address the challenge of limited labeled data [75]. These techniques allow models to leverage knowledge from related tasks or domains, thereby improving their performance in scenarios where labeled data is scarce.

Furthermore, the integration of attention mechanisms into segmentation models has been shown to enhance the model's ability to focus on relevant regions of the image, leading to more accurate segmentations [47]. However, these models still require substantial computational resources and large amounts of data, which can be a limiting factor in practical applications.

In summary, while semantic segmentation offers significant advantages in terms of localizing objects within an image and improving explainability, it is

also associated with challenges related to data requirements, generalization, and interpretability. Ongoing research continues to explore new architectures and techniques to overcome these challenges, with the goal of making semantic segmentation more robust and accessible in real-world applications.

Chapter 8

Out-of-Distribution Anomaly Detection

Out-Of-Distribution Anomaly Detection (OOD-AD) is a vital aspect of ensuring the safety and reliability of Machine Learning (ML) systems, especially in applications where the consequences of errors can be severe [18]. OOD-AD plays a crucial role in a wide range of domains, from autonomous driving to medical diagnosis, where the detection of anomalies—instances that deviate from the expected distribution of data—is essential to prevent catastrophic failures [67].

In the context of this thesis, we specifically focus on the application of OOD-AD within the domain of pharmaceutical Quality Control (QC), a critical area where the integrity and safety of products must be rigorously maintained [49]. Visual Inspection (VI) systems powered by Computer Vision (CV) techniques have become indispensable in this industry, enabling automated, accurate, and efficient quality checks [114]. VI systems in the pharmaceutical industry are designed to perform *non-contact*, *non-destructive* evaluation of products during the manufacturing process. These systems utilize advanced imaging technologies to capture high-resolution images of pharmaceutical products, which are then analyzed using CV algorithms [161]. The goal is to detect defects, contaminants, or any deviations from the specified quality standards [28].

CV, a subfield of AI, involves the development of algorithms that can interpret visual information from the world. In the pharmaceutical industry,

this technology is used to ensure that products meet stringent regulatory requirements. For example, VI systems can be used to check the integrity of packaging, the correctness of labeling, and the consistency of tablet shapes and colors [154].

The importance of OOD-AD in this context cannot be overstated. During the manufacturing process, unexpected anomalies can occur due to various factors, such as changes in the raw materials, equipment malfunctions, or environmental variations. Detecting these Out-Of-Distribution (OOD) anomalies is essential to prevent defective products from reaching the market, which could lead to significant health risks and financial losses.

8.1 Overview of the Methods

One of the main challenges in industrial quality control is dealing with highly imbalanced datasets, where there is no comprehensive knowledge of potential defects. As previously mentioned, the defect classes typically encountered are those collected from production line outputs and manually classified by operators conducting exhaustive inspections. Generally, the problem in OOD-AD is to identify defects that deviate from "normality." To define this normality, it is first necessary to statistically characterize the problem. According to the law of large numbers, the features of products will follow a Gaussian distribution (Equation 8.1), where the population of good products will cluster around the mean, while defects or outliers will deviate beyond a certain threshold. In industrial settings, particularly in Industry 4.0, similar approaches have already been observed, where an outlier is defined as a value that falls outside the range $\mu \pm 3\sigma$ [146].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{8.1}$$

This approach can be defined explicitly or implicitly, where in the latter case, the network learns to reconstruct or compress only the "normal" features while failing to reconstruct or compress the "anomalous" ones.

In this chapter, three approaches will be described. The first two are

reconstruction-based, as described in the works GRD-Net: Generative Reconstructive Discriminative Anomaly Detection with Region of Interest Attention Module [43] and Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line [44]. In these works, inspired by the research in GANomaly [2], a deep network—enhanced with residual blocks in the described case—is trained to reconstruct the product. Noise is added to the product image to occlude parts of the image, making the network's reconstruction capabilities more robust. The products analyzed in both cases are two different strips of vials containing a liquid active ingredient that, due to its fluid dynamics properties, closely resembles distilled water, while the container is made of BFS. The two works differ mainly in the second part, the discriminative one, where a heatmap is generated, and the image is subsequently segmented to satisfy the requirement for explainability of the results provided by the Neural Network (NN). In the first approach, a method similar to that used in DRÆM is proposed, where a second network generates a heatmap from the input and generated images, with the addition of a Region of Interest (ROI) during training, highlighting the area where the defect search should be concentrated. In the second case, the approach is different due to the much tighter timing constraints between acquisition and the trigger that physically rejects the product on the conveyor belt within the machine, so the approach is simplified.

The last study that will be presented is still in development on a product protected by NDAs, which do not allow for images to be shown. It involves syringes containing a two-component active ingredient in powder form. Due to its much noisier nature compared to the liquid, the reconstruction network struggles to find a "rule" and therefore to generalize it to reconstruct the product, resulting in many errors and generating numerous false rejects. The approach under study is instead based on embedding similarity systems, where the salient features are compressed by a network pre-trained on a large, diverse dataset (e.g., ImageNet [139, 88] and its derivatives). These features are then analyzed and classified based on spatial properties (e.g., representing them in an $\mathbb{R}^N$ space, where N is the number of features for each embedding of each extracted patch) and calculating the anomaly score based on the Euclidean distance in this space as in PatchCore [137], or by addressing it probabilistically,

using tools such as the Mahalanobis distance (Equation 8.2), as in PaDiM [32], or using the concept of probability flow [135] (Equation 8.3) as in FastFlow [168].

$$D_M(x) = \sqrt{(x-\mu)^\top \Sigma^{-1}(x-\mu)} \tag{8.2}$$

$$\frac{\partial p(x,t)}{\partial t} = -\nabla \cdot [\mathbf{v}(x,t)p(x,t)] + D\nabla^2 p(x,t) \tag{8.3}$$

In summary, the three main methods utilized in this work are:

- **GRD-Net**: A reconstruction-based method that integrates generative, reconstructive, and discriminative components with a Region of Interest Attention Module. This method focuses on reconstructing the product image and emphasizes regions prone to defects, making it particularly effective in high-stakes environments like pharmaceutical quality control [43].
- **Deep Generative Anomaly Detection with Residual Blocks**: Another reconstruction-based approach that utilizes deep networks enhanced with residual blocks. This method involves adding noise to the product image to improve the network's reconstruction capabilities, making it robust to occlusions and minor defects [44].
- **Embedding Similarity Based on PatchCore**: This method approximates the PatchCore [137] approach by using embedding similarity systems. It compresses salient features using pre-trained networks and calculates anomaly scores based on Euclidean distance and Mahalanobis distance. This hybrid method addresses the challenges of memory-intensive methods by approximating the memory bank with online computations.

8.2 Reconstruction-based Methods for Out-of-Distribution Anomaly Detection in Computer Vision

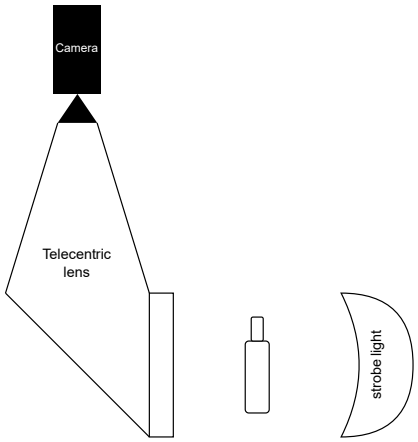
Reconstruction-based methods in semi-supervised networks are distinguished by their ability to learn and generalize from a limited set of labeled data, typically the normal cases in AD scenarios. These methods primarily utilize architectures such as Autoencoders (AEs) [7, 110], Variational Autoencoders (VAEs) [86, 84], and Generative Adversarial Networks (GANs) [56] to reconstruct normal operational data and thereby identify anomalies when deviations occur.

AEs are particularly effective for learning a compressed representation of data, enabling the model to detect discrepancies during the reconstruction phase, which signals the presence of anomalies [10]. On the other hand, GANs, through architectures like GANomaly [2], employ a generative approach that facilitates robust reconstruction, employing residual blocks within fully convolutional networks to prevent gradient vanishing and thus enhancing model stability [164]. The integration of a Region of Interest (ROI)-based attention module represents an innovative step forward, refining the focus on specific areas during training to improve the detection and segmentation of anomalies [170, 43].

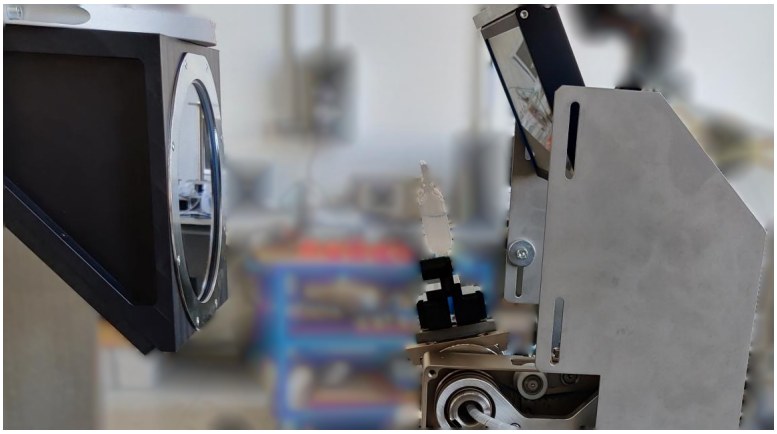
However, challenges remain, particularly concerning the setting of thresholds for AD, which can be prone to errors in noisy data conditions. Addressing these challenges is crucial for future advancements, aiming to enhance the specificity and reduce false positives in industrial applications.

8.2.1 Problem Description

The study examines two related scenarios involving the inspection of five BFS vials per strip. The second scenario presents a particular challenge due to the curved and thinner walls of the vials, complicating both the superficial inspections and content analysis. Each inspection station, equipped with a ML algorithm, faces forward and uses a telecentric lens to minimize perspective distortion, with a similar setup on the opposite side.



(a) Camera1: station scheme



(b) Camera1: station design in lab

Figure 8.1: (a) Scheme of the acquisition station. (b) Laboratory-built sketch to test lighting and framing.

As depicted in Figure 8.1, the setups are identical for both scenarios. The first allows 1 second of inference time (2 machine cycles), suitable for employing GRD-Net. During this phase, data processing is handled sequentially by an AE and a U-Net, generating a heatmap. Conversely, the second scenario's limited 500 ms time frame necessitates a simplified architecture by omitting the AnoSeg network. Enhancements to the ResAE [164, 177, 104] include inspirations from WRN [169], used in classification, and revisions to the contextual loss functions.

Detecting defects is crucial as they typically occupy a minimal area on the surface. For example, common defects in pharmaceutical vials include minor scratches, small black spots, or foreign particles, ranging from 100 to 1000 µm.

Additionally, GRD-Net [43] aims to lessen the dependence on preprocessing algorithms traditionally employed for AD. Unlike standard reconstruction-based methods that rely on threshold-based classifiers for anomaly mapping—often at the expense of specific ROIs focus—the embedding similarity-based approach promises enhanced results. Nevertheless, this method faces hurdles in achieving clear outcomes and a streamlined learning process, which complicates the precise targeting of ROIs to accentuate defects.

8.2.2 Method

To explain the mentioned approaches, this section introduces GANomaly [2] and DRÆM [170] as they are the foundational knowledge necessary for understanding the rest of the chapter.

8.2.2.1 GANomaly

8.2.2.1.1 Adversarial Autoencoders An AE, as defined by [55], is a neural network trained to replicate its input at its output. It comprises an encoder (E) that maps inputs into a latent space h, and a decoder (D) that reconstructs the input from h. The encoder-decoder configuration is especially potent when h is dimensionally smaller than x, making the AE undercomplete. This design compels the network to capture the most salient features of the input data, a process optimized by minimizing the penalty function of the network when outputs diverge from x. Introducing adversarial training en-

vironments enhances the AE's performance by improving reconstruction and allowing finer control over the latent space [29, 107, 111].

8.2.2.1.2 Generative Adversarial Networks GANs, developed as an unsupervised learning method primarily to generate realistic synthetic images, employ two competing networks: a generator and a discriminator [56]. The generator, functioning akin to a decoder, learns the input data distribution from a latent space, whereas the discriminator assesses the authenticity of images produced by the generator.

8.2.2.1.3 GANomaly Architecture and Training The GANomaly architecture utilizes a novel encoder-decoder-encoder structure alongside discriminator networks [2]. It starts with an AE acting as the generator, which reconstructs the input image x into $\hat{x}$. A second encoder then compresses $\hat{x}$, mirroring the first encoder's architecture but with differing parameters, thus learning to minimize the parametric distance, which is crucial for AD during testing.

Training of GANomaly involves minimizing a compound loss consisting of adversarial, contextual, and encoder components. The adversarial loss $\mathcal{L}_{adv}$ enhances training stability. Contextual loss $\mathcal{L}_{con}$ integrates contextual information into the loss and is defined as:

$$\mathcal{L}_{con} = \omega_a \mathcal{L}_1(x, \hat{x}) + \omega_b(1 - \text{SSIM}(x, \hat{x})). \tag{8.4}$$

where $\mathcal{L}_1$ is the $\mathcal{L}_1$ norm loss also known as Mean Absolute Error (MAE) loss, measuring pixel-wise difference between x and $\hat{x}$, and SSIM is the Structural Similarity Index, which evaluates perceived quality similarity. The parameters ω_a and ω_b are weights that balance the contributions of $\mathcal{L}_1$ and SSIM terms to the total contextual loss.

Encoder loss $\mathcal{L}_{enc}$ minimizes the discrepancy between the bottleneck features of x and the encoded features of $\hat{x}$. The total loss is then expressed as:

$$\mathcal{L}_{gan} = \omega_1 \mathcal{L}_{adv} + \omega_2 \mathcal{L}_{con} + \omega_3 \mathcal{L}_{enc}. \tag{8.5}$$

The weights ω_1, ω_2, and ω_3 adjust the influence of each component on the total loss. Empirical adjustments from initial parameters proposed in the foundational papers of GANomaly [2] led to:

$$\begin{aligned} \omega_a &= 1.0 \\ \omega_b &= 1.0 \\ \omega_1 &= 1.0 \\ \omega_2 &= 50.0 \\ \omega_3 &= 1.0. \end{aligned} \tag{8.6}$$

Using a branch and bound method, adjustments in steps of ± 5 for each ω_* while holding others constant showed that increasing ω_2 to 50 optimizes training duration without compromising the loss components' effectiveness.

8.2.2.2 Residual Autoencoder

Conventional Autoencoders (AEs) often suffer from the vanishing gradient problem, a prevalent issue also observed in Deep Convolutional Neural Networks (DCNNs). This challenge complicates the training process as it impedes the effective propagation of gradients through the network during backpropagation, especially in deeper architectures. To address these challenges, Residual Networks (ResNets) have been extensively studied and implemented, as they introduce residual connections that facilitate the flow of gradients and stabilize training [64]. While ResNets are predominantly used in supervised learning environments for classification tasks, their underlying principles have been adapted to enhance AEs. The introduction of residual blocks within Residual Autoencoders (ResAEs) has expanded their applicability to unsupervised learning scenarios. Recent studies demonstrate that ResAEs not only preserve the structural integrity of data more effectively but also improve AD and feature representation capabilities in various applications [43, 164, 178, 177, 90]. A schema of one residual stage composed of two residual blocks is shown in Figure 8.2.

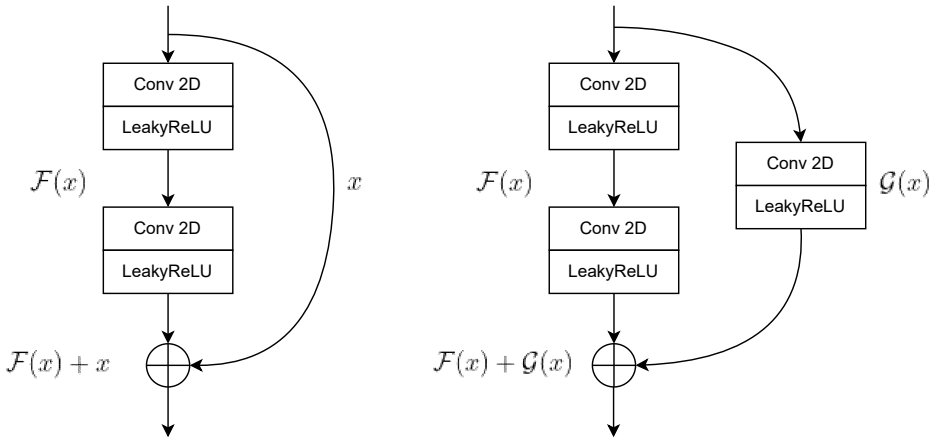
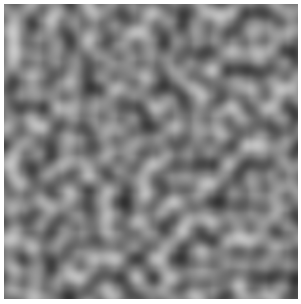
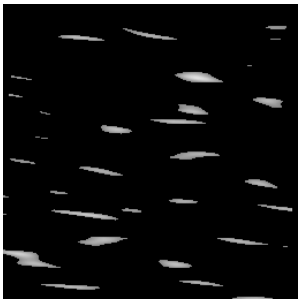


Figure 8.2: Two consecutive residual blocks of one stage of the encoder network. The introduction of a residual architecture in the encoder-decoder-encoder GAN [64] revealed to be more stable during training phase, by giving better results with equal epochs. First residual block (on the left) maintain the same output size, using a kernel of (3×3) and a stride of (1×1), the second one (on the right) reduce halve H and W sizes, using a kernel of (3×3) and a stride of (2×2), so a Conv2D with stride of (2×2) or a combination of a Conv2D with a stride of (1×1) and a Pooling Layer is needed to consistently combine the outputs.

8.2.2.2.1 Wide Residual Autoencoder Expanding upon the standard ResAE, Wide Residual Autoencoders (WRAEs) incorporate the architectural principles of Wide Residual Network (WRN), which involve increasing the width of the network rather than its depth. This adjustment has been shown to enhance the model's ability to learn more complex features without a significant increase in computational complexity [169]. Has been explored the adaptation of WRN's broadened architecture in ResAEs in *Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line* [44], where ResGANAD is implemented, highlighting improvements in efficiency and performance in unsupervised tasks. These modifications not only reduce the impact of the vanishing gradient problem by shortening the path that gradients need to traverse but also enhance the network's capability in handling more complex and diverse datasets. Implementations of WRAEs have demonstrated promising results in fields like image reconstruction, denoising, and AD, further broadening the scope of applications for residual learning in unsupervised environments [90, 177].



(a) Perlin Noise



(b) Perlin Noise with random RGB pixels

(c) Dataset's image with Perlin noise

Figure 8.3: Simulated anomaly generation process. 8.3a An example of Perlin noise. 8.3b Represents the merging of the anomaly map and random RGB pixels. 8.3c Represents an example of an image with generated fake anomalies.

8.2.2.3 DRÆM

As mentioned in Section 5.3.3, Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection (DRÆM) is an AD framework based on two different sub-networks. The first sub-network (called reconstructive sub-network), is trained to recognize anomalies and reconstruct them while keeping the portions of the input image that are not anomalous. The second network learns joint anomaly-inclusion reconstruction to create accurate anomaly segmentation maps by fusing the original and reconstructed appearance.

Instead of generating simulations that accurately reflect the actual appearance of the anomaly in the target domain, DRÆM instead creates just-out-of-distribution appearances that allow learning the proper distance function to identify the anomaly by its departure from normality. This paradigm is used in the proposed anomaly simulator. The images with artificial anomalies are generated through Perlin noise generator [124] to generate a variety of anomaly shapes (see Figure 8.3a). The generated image, is then binarized by a threshold into an anomaly map using uniform random samples. Then, merging the anomaly map with random RGB pixels, we obtain the final noise (see Figure 8.3b) to be added to the images of the dataset as can be seen in Figure 8.3c. Thus, this process creates training sample triplets with the original image that is free of anomalies, the augmented image that contains simulated anomalies, and the pixel-perfect anomaly mask.

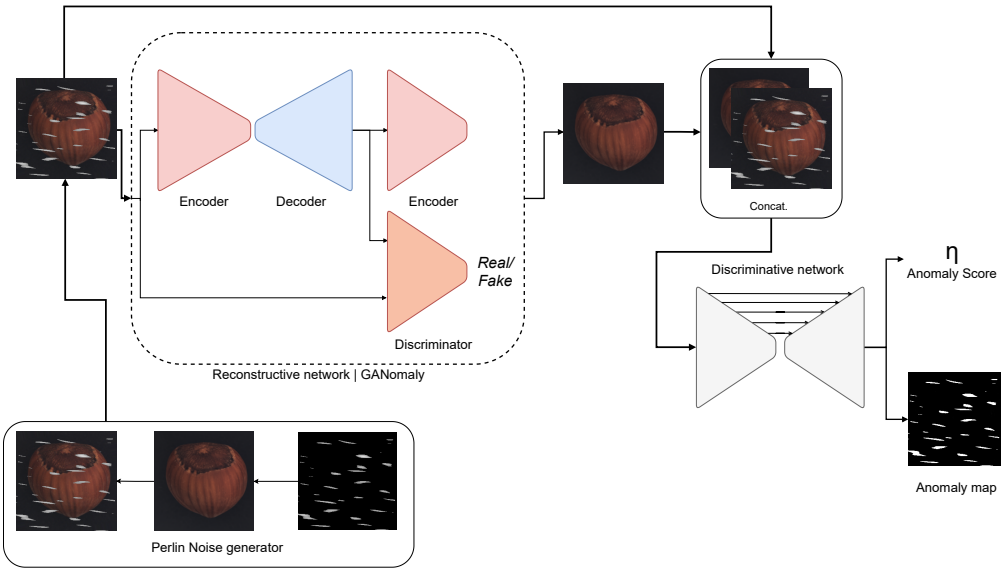


Figure 8.4: The architecture of GRD-Net. This architecture mirrors that of the vanilla DRÆM but implements GANomaly in place of the ResAE, which served as the Reconstructive network.

The reconstructive sub-network of DRÆM perform an image denoising task. It is trained to reconstruct the original image from the artificial corrupted version produced by the process described above. The discriminative sub-network is a U-Net-like neural network that take in input the channel-wise concatenation of the reconstructive sub-net output and the original image. This second sub-network learns to segment the Perlin noise applied to the original image instead to used a similarity functions such as SSIM [163].

The output of the discriminative sub-network is an AD mask. This mask can be interpreted for the image-level anomaly score estimation. The anomaly mask is smoothed by a convolutional filter. The final anomaly score is computed by taking the maximum value of the smoothed anomaly score map.

8.2.2.4 GRD-Net: Generative-Reconstructive-Discriminative Network with Attention Module

GRD-Net [43] is heavily inspired by DRÆM [170], as is evident in the general architecture of the proposed framework. As illustrated in Figure 8.4, the setup closely resembles the vanilla DRÆM; however, it substitutes the standard AE with GANomaly for the Reconstructive network. To preserve the

structural integrity and prevent training degradation, all networks within the reconstructive sub-network are designed with residual architectures.

The training of the GANomaly within the GRD-Net utilizes the loss described in Equation 8.5. Additionally, the discriminative network employs Focal Loss (FL) [96, 170], defined as follows:

$$\mathcal{FL}(p) = -(1-p)^{\gamma}\log(p). \tag{8.7}$$

$\mathcal{FL}$ introduces the term $-(1-p)^{\gamma}$ to the conventional cross-entropy loss, where setting $\gamma > 0$ focuses more on misclassified examples [96]. This selective focus enhances the discriminative network's accuracy, especially in segmenting challenging examples. To ensure that defects on the surfaces of inspected products are precisely identified, a segmentation mask defining the Region of Interest (ROI) is utilized in conjunction with the AD mask. This combination yields an *intersection mask*, calculated as follows:

$$\mathcal{I} = \mathcal{A}_{discr} \times \mathcal{ROI}_{input}. \tag{8.8}$$

Here, $\mathcal{I}$ represents the intersection mask, a tensor resulting from multiplying the input mask tensor $\mathcal{ROI}_{input}$, which highlights the ROI, with the output mask tensor $\mathcal{A}_{discr}$ of the discriminative network.

$$\mathcal{L}_{tot} = \mathcal{L}_{gan} + \mathcal{FL}(\mathcal{I}, \mathcal{M}_{input}). \tag{8.9}$$

The total loss function $\mathcal{L}_{tot}$ combines the GAN Loss $\mathcal{L}_{gan}$ with the FL calculated over the intersection area $\mathcal{FL}(\mathcal{I})$.

The training and inference processes are detailed in Figures 8.5 and 8.6, respectively.

8.2.2.4.1 ROI Attention Module The Zipper dataset was used to evaluate the Attention Module providing the ROI. This dataset is particularly suitable as it contains two logical regions of interest: the zipper area itself

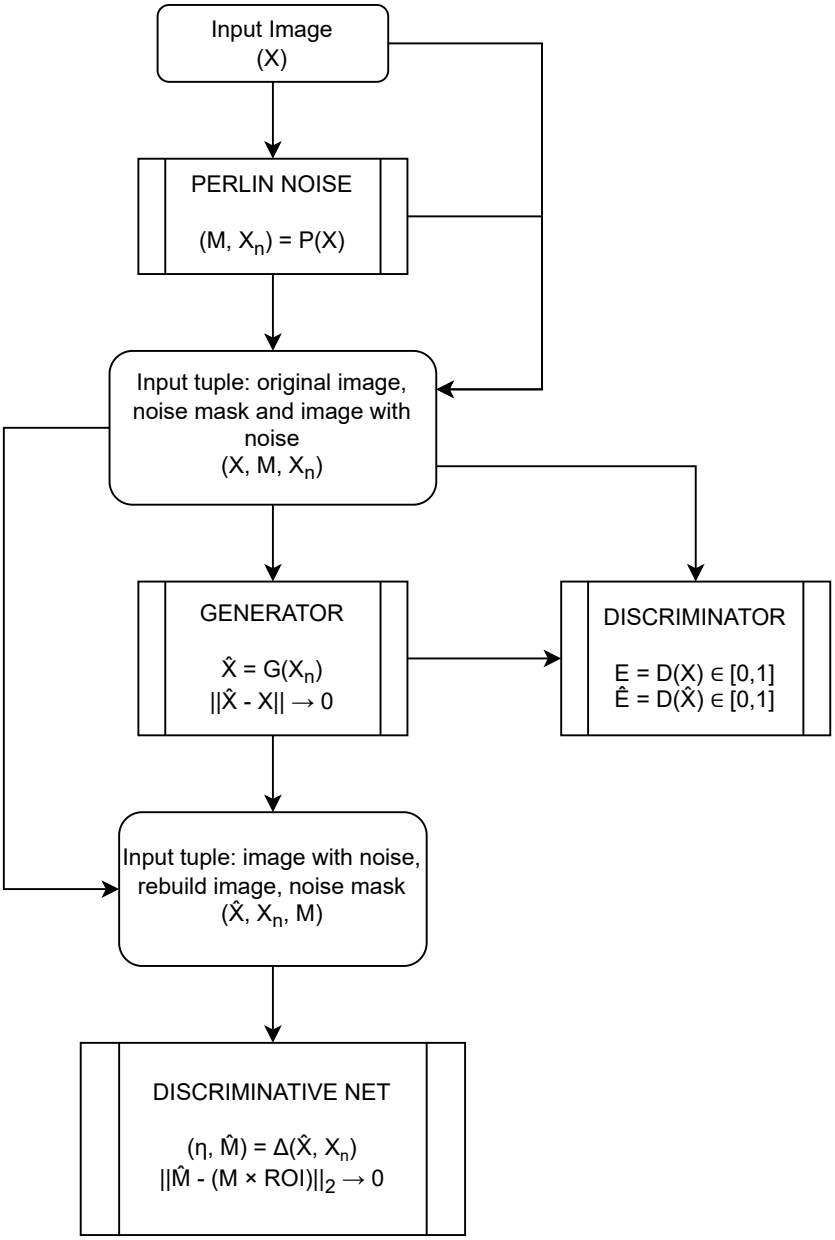


Figure 8.5: Training step flowchart: Input image X is transformed into X_n, which is the image with superimposed Perlin noise. M represents the mask of the noise areas.

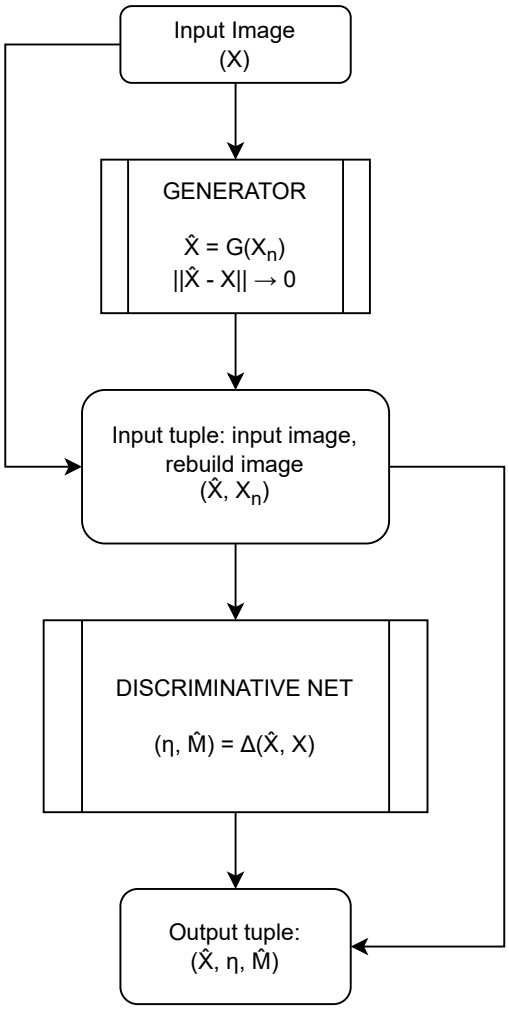


Figure 8.6: Inference step flowchart.

and the fabric area. In this experiment, the region of interest was defined as the zipper, thus excluding defects in the fabric zone. An example is shown in Figure 8.7.

As mentioned earlier, the *discriminative* network was trained using the focal loss applied to the intersection between the ROI and the mask generated from Perlin noise. This ensures that the network learns not only to detect anomalies by comparing original and reconstructed images but also to focus on the area where these anomalies are likely to appear.

In most industrial cases, not all regions of the image are relevant for detecting defects. In fact, irrelevant regions can sometimes be misleading, as they may contain anomalies that are outside the scope of the inspected product.

8.2.2.5 ResGANAD

8.2.2.5.1 Network The network proposed herein utilizes an encoder-decoder-encoder configuration for training, featuring a generator. The encoder employs a Wide Residual Autoencoder (WRAE) architecture as depicted in Figure 8.8, while the decoder, illustrated in Figure 8.9, adopts a reverse residual structure. As discussed in section 8.2.2.2, ResNets and ResAEs [64, 65, 43, 164, 178, 177, 90, 44] are prevalently utilized due to their efficacy in mitigating training degradation issues such as gradient vanishing. This work advances the residual architecture to include more recent developments, initially based on designs from previous studies [43] and inspired by the innovations by Zagoruyko et al. [169].

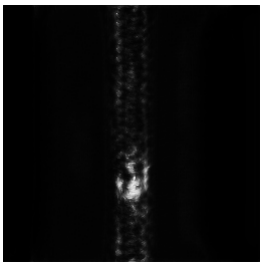
8.2.2.5.2 Application specific optimizations Building upon the foundational concepts outlined in section 8.2.2.4, this section delves into application-specific enhancements. We employed various augmentation techniques such as adding perturbations and Perlin noise, alongside other methods including random rotations within the range of $[-\frac{\pi}{8}, \frac{\pi}{8}]$ radians and random vertical flips. Note that horizontal flips and rotations greater than $\frac{\pi}{2}$ are treated as anomalies.

These augmentation techniques bolster the network's generalization capabilities but also introduce significant entropy into the training process, potentially leading to instability or divergence. Consequently, a *Noise Loss* was

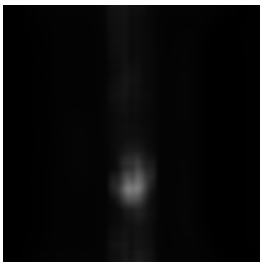
(a) Original image (X)

(b) Reconstructed image by the Generator G ($\hat{X}$)

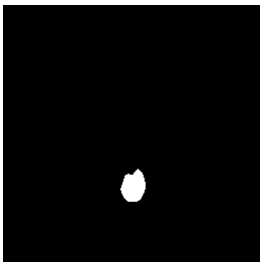
(c) Ground truth (M)



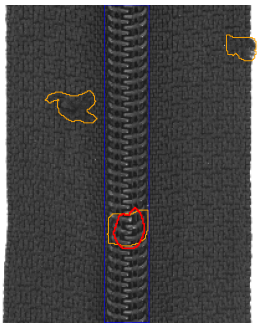
(d) Generated *heatmap* by *discriminative* model



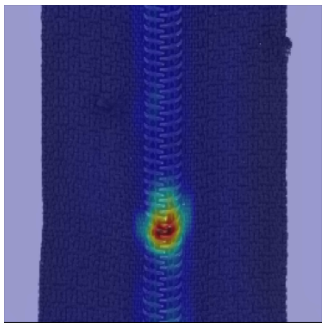
(e) Generated *heatmap* by *discriminative* model after average pooling with 21×21 kernel



(f) Resulting anomaly localization region ($\hat{M}$)

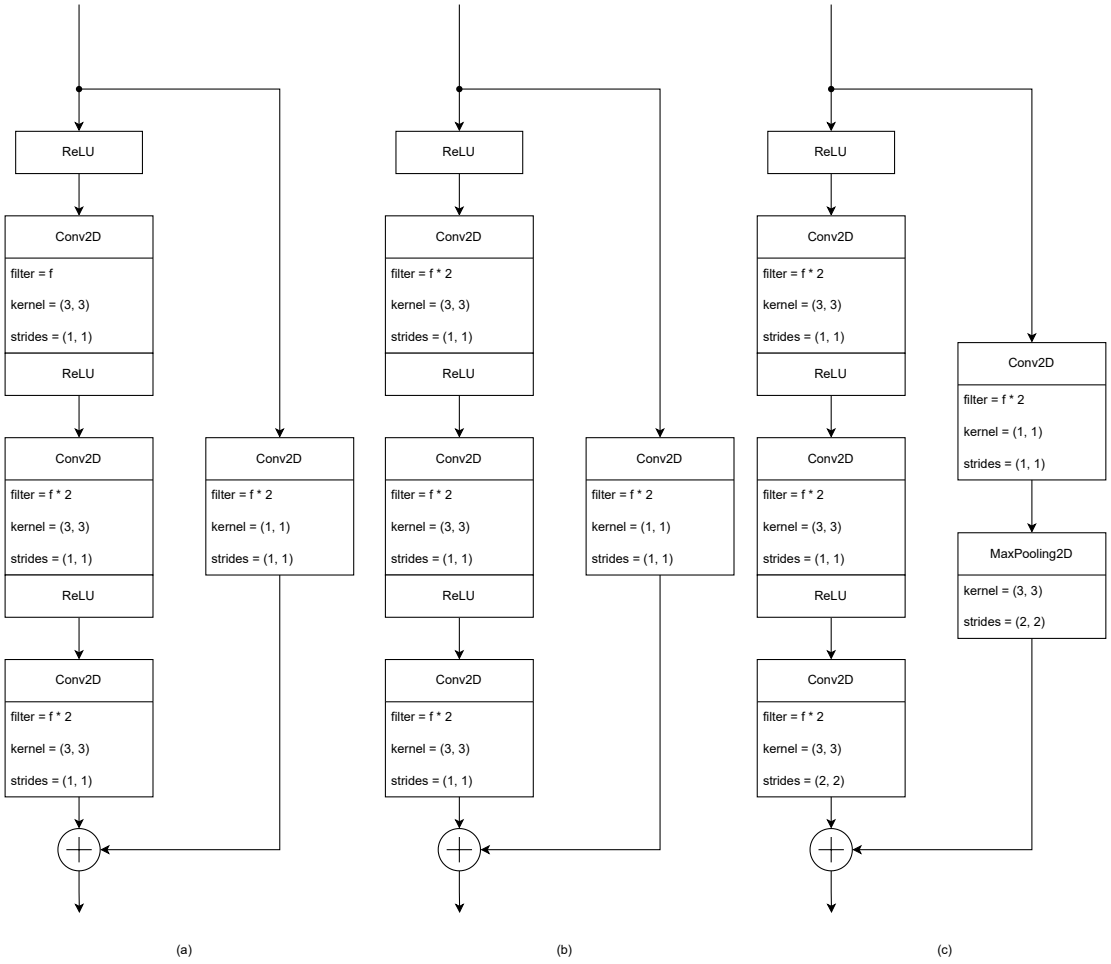


(g) Original image with regions: **blue** region is the *ROI*; the **orange** region is the ground truth (M); and the **red** region is the generated region ($\hat{M}$)



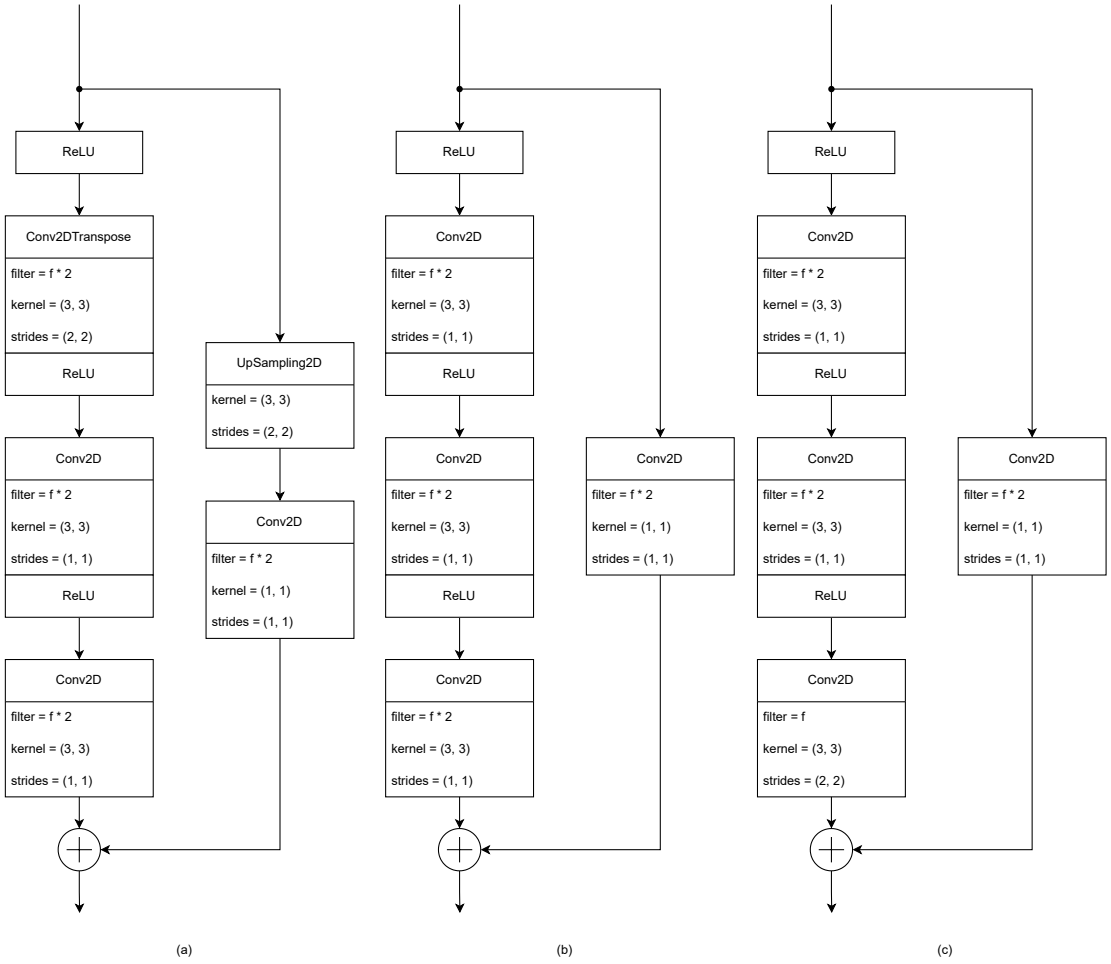
(h) Image generated by superimposing the convoluted heatmap from the *discriminative* net onto X, and colorizing it with *jet* colormap

Figure 8.7: An example from the zipper dataset showing three anomalies: one in the zipper, one in the middle of the fabric part, and another on the border of the fabric zone. Only the anomaly within the zipper region (inside the ROI) is detected, and it is almost perfectly aligned with the ground truth defect region.



(a) Residual encoder

Figure 8.8: Configuration of three residual blocks in the encoder, where only the final block reduces the (H, W) dimensions of the layer. Following the initial convolutional layer, the number of filters doubles: filters $= f \cdot 2$. The typical sequence in a residual stage is (a)-(b)-(c).



(a) Residual decoder

Figure 8.9: Configuration of three residual blocks in the decoder, where only the final block increases the (H, W) dimensions of the layer. With the exception of the last convolutional layer, the number of filters doubles: filters $= f \cdot 2$. The sequence for a residual stage is (a)-(b)-(c).

introduced:

$$\mathcal{L}_{nse} = w_4 \cdot \mathcal{L}_2(|(1.0 - \beta)\hat{M} \cdot \hat{X} - M \cdot \overset{*}{\hat{X}}|, \beta N), \tag{8.10}$$

where the term $\overset{*}{X}$ represents the input post-perturbation and is defined by the following equation, generated by the function $P_q(X)$:

$$\begin{aligned}
&\overset{*}{X} = (1 - M) \cdot X + (1.0 - \beta)M \cdot X + \beta N, \\
&\beta \sim \mathcal{U}(0.5, 1.0), \\
&N = \text{Perlin noise isolated}, \\
&M = \begin{cases} 1, & \text{if the pixel belongs to the noise region } N, \\ 0, & \text{otherwise} \end{cases}
\end{aligned} \tag{8.11}$$

where $\overset{*}{X}, M, N, \beta = P_q(X)$ represents the input post-perturbation, and N denotes the isolated Perlin noise applied to a black image of the same size as X. Equation 8.11 illustrates how the noise is superimposed onto X in P_q. The parameter q represents the probability of applying Perlin noise to the image X, with $1 - q$ being the probability that $\overset{*}{X} = X$, $M = \mathbb{0}$, and $N = \mathbb{0}$. The Perlin noise not only introduces non-Gaussian noise—more closely resembling real defects—to enhance the network's denoising capabilities but also serves to mask a patch [62, 126], which, in this case, is non-rectangular, compelling the model to accurately learn the characteristic features of a product defined as *good*. This approach helps to counteract a typical weakness of vanilla autoencoders, which often replicate even minor defects, especially when they are very small. Empirical evidence suggests that this method enhances reconstruction accuracy beneath the noise overlay.

Additionally, the l1-norm loss previously used in the contextual loss was replaced with a Huber loss [52], which mitigates the non-differentiability at the origin, as follows:

$$\mathcal{L}_{con} = w_a \cdot \mathcal{L}_{Huber}(X, \hat{X}) + w_b \cdot \mathcal{L}_{\text{SSIM}}(X, \hat{X}), \tag{8.12}$$

Drawing from GANomaly [2], and through a branch-and-bound analysis,

the optimal parameter configuration for this application was determined as:

$$\begin{aligned} w_a &= 2.0 \\ w_b &= 1.0 \\ w_1 &= 1.0 \\ w_2 &= 50.0 \\ w_3 &= 1.0 \\ w_4 &= 3.0, \end{aligned} \tag{8.13}$$

Despite $\mathcal{L}_{\text{SSIM}}$ playing a critical role in the reconstruction of the input image, it tends to become unstable with complex, high-entropy images. Thus, increasing w_a moderated this issue at the expense of prolonged convergence times.

8.2.2.5.3 Classification and Segmentation This application does not necessitate segmentation; the generation of a heatmap and identification of anomalous patches suffice to meet the project's objectives. Consequently, a generative network alone was deployed during both training and inference phases. Anomalies are quantified by the score ϕ:

$$\phi = 1 - \text{SSIM}(X, \hat{X}), \tag{8.14}$$

The heatmap is derived from the absolute difference between the input and the output, normalized between 0 and 1:

$$H = |X - \hat{X}|\Big|_0^1, \tag{8.15}$$

The normalization formula is expressed as:

$$H = a + b \cdot \frac{|X - \hat{X}| - \min(|X - \hat{X}|)}{\max(|X - \hat{X}|) - \min(|X - \hat{X}|)},\ a = 0 \wedge b = 1. \tag{8.16}$$

The optimal threshold ϕ_t maximizes detection accuracy on both standard and anomalous real samples, based on a subset of real production data (both positive and negative). A patch classified as anomalous during inference implies rejection of the entire product, and in such cases, 8.15 is employed to

generate a heatmap for the corresponding anomalous patch.

8.2.3 Approach Limitations

The principal limitations of this technology stem from the requirements for extensive datasets or substantial data augmentation to effectively train models capable of reconstructing and potentially segmenting the input. The challenge of securing large datasets is often compounded by limited availability from clients who manufacture the products. Furthermore, data augmentation must avoid overfitting while aligning with acceptable product standards and client-specific defects, a balance that is not easily achieved. Overly aggressive augmentation can introduce non-representative edge cases or defects, potentially confusing the network, while insufficient augmentation may lead to overfitting.

Another significant concern is the size of the model and the consequent computational load of the networks that constitute the architecture. This creates a trade-off between performance and the practical constraints of processing time and hardware availability, often linked to cost considerations.

Moreover, while models generally achieve higher accuracy with larger datasets, in situations marked by data scarcity or prohibitive computational costs, systems based on embedding similarity are frequently preferred for their reduced computational requirements.

8.2.4 Related Work

GRD-Net [43], is inspired by the well-established DRÆM [170]. This model employs the *reconstruction-based* methodology prevalent in surface anomaly detection, aiming to exploit the inability to faithfully reconstruct out-of-distribution regions within an image [108, 166]. Commonly utilized neural networks in this domain include AEs, VAEs, and GANs [2, 3, 16, 54, 160, 125, 140, 166], with anomaly detection hinging on the quality of image reconstruction, assessed via structural similarity [16] or pixel-wise reconstruction errors [14]. Visual attention maps derived from the latent space also contribute to anomaly mapping [160]. Despite their interpretability, the efficacy of reconstruction-based methods is occasionally hindered by their limited ability to differentiate

anomalous images effectively [123]. Additionally, advancements in transformer technology have enabled enhanced reconstruction and segmentation capabilities [167, 112, 41]. The network's interpretability and the constrained size of the latent space, despite a computationally intensive architecture, represent significant advantages. The integration of contrastive learning techniques has also shown promising improvements in this field [156].

Reconstruction-based AD has also been applied to varied data types like time series, where traditional methods prove inadequate [94]. Integration of LSTM-RNN models into GAN or VAE-GAN architectures addresses these challenges, often arising from industrial processes or smart grids [174, 119, 78, 128, 148, 20].

The influence of Zavrtanik et al. [170] is particularly noteworthy, as their model, DRÆM, incorporates a reconstruction and a discriminative network to segment artificial noise, producing an AD mask and anomaly score, thereby facilitating the estimation of the image-level anomaly score from the maximum value of the smoothed anomaly score map.

8.3 Embedding Similarity-based Methods for Out-of-Distribution Anomaly Detection in Computer Vision

Embedding Similarity-based Methods are the other significant family of OOD algorithms for AD. These methods operate on the principle that anomalous data points, which do not belong to the distribution of normal data, will manifest as outliers when mapped into a lower-dimensional embedding space. This space is typically learned by a Deep Neural Network (DNN), often pre-trained on a large dataset (e.g. ImageNet), where the features of the input data are transformed into a compact representation.

The key idea behind these methods is that in the embedding space, the distance or similarity between the embeddings of in-distribution (normal) data points will be small, while the distance between the embeddings of OOD (anomalous) data points and normal data points will be significantly larger. Various distance metrics, such as Euclidean distance, cosine similarity, or more

sophisticated measures like the Mahalanobis distance, are employed to quantify these differences [97, 66, 93].

One widely used approach in this category is the Mahalanobis-based method, where the Mahalanobis distance is computed between the test sample's embedding and the mean of the embeddings of the training samples. If this distance exceeds a certain threshold, the sample is classified as anomalous. This method was popularized by the work of Lee et al. [93], where they demonstrated its effectiveness in detecting OOD samples across various computer vision tasks. Another notable work that utilizes this metric to detect and segment anomalous images is PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization, proposed by Defard et al. [32]. PaDiM applies the Mahalanobis distance in a patch-wise manner across the image, modeling the distribution of feature embeddings for each patch. This method is particularly effective in identifying and localizing anomalies within an image, as it captures the spatial distribution of features, thereby enabling more precise anomaly detection and segmentation.

Another notable technique is the use of contrastive learning, where a model is trained to maximize the similarity between embeddings of similar (in-distribution) samples while minimizing the similarity between embeddings of dissimilar (OOD) samples. The SimCLR framework [22] and its variants have been employed to enhance the discriminative power of embeddings, making them more robust in distinguishing between normal and anomalous data.

In addition to contrastive learning, other approaches leverage self-supervised learning tasks to improve the quality of embeddings. For example, Golan and El-Yaniv [53] proposed an approach where the model is trained to solve a jigsaw puzzle or predict image rotations as a pretext task. The underlying assumption is that the model will learn to extract meaningful features from the data, which can then be used to identify OOD samples by evaluating the similarity of the resulting embeddings.

Recent advances have also explored the combination of embedding similarity with probabilistic models. For instance, the work of Ren et al. [132] combines an energy-based model with an embedding similarity measure to better capture the uncertainty associated with OOD samples, thus improving the overall detection performance.

8.3.1 Problem Description

The study presented in this section is still in its preliminary stages, as we are currently addressing several structural limitations of generative model-based approaches, such as GANs and AEs, in the context of OOD-AD. To date, our methodology has only been tested on benchmark datasets like MVTec, which, while not fully representative of industrial processes or the complexity of the products manufactured in such settings, serve as a solid foundation for analyzing the problem and benchmarking potential solutions.

One of the primary challenges with generative models is the significant requirement for large datasets of "regular" products, i.e., non-defective items. In order to efficiently learn and compress the salient features of the source images, these models necessitate a large quantity of training examples. On smaller datasets like MVTec, generative-based AD models tend to underperform, primarily due to the need for extensive data augmentation, which is often impractical. For example, certain transformation functions used during augmentation might inadvertently alter a non-defective sample into one that appears defective, complicating the training process by introducing spurious examples.

Another notable weakness of generative models is their inference speed. Architectures such as AEs or, more generally, U-Nets are computationally expensive both in the training and inference stages, requiring not only longer processing times but also more powerful hardware.

A third limitation, particularly associated with *vanilla* AEs or GANs based on AEs, is their tendency to "copy" not only the main features of the input but also the defect itself. This behavior can be highly unstable and is often sensitive to the size of the latent space chosen. However, this issue has been largely addressed in recent works [170, 43, 44, 62], where reconstruction extends beyond simple replication to include denoising or masked patch reconstruction, as discussed in Section 8.2.2.5.2, forcing the network to remove noise that does not belong to the distribution of regular features and to reconstruct beyond these patches, generalizing from the surrounding features.

One key advantage of embedding similarity-based methods is that they often rely on pre-trained backbone networks on large-scale and diverse datasets like ImageNet [139], eliminating the need for extensive retraining or fine-

tuning. Nevertheless, these approaches have their own limitations. A major drawback is the low expressivity of the backbone network, which, being unfamiliar with the specific products and contextual features of the current dataset, may fail to accurately capture the relevant features for the domain in question.

Several solutions have been proposed to address this limitation. One approach involves introducing a linear layer [100] that acts as a feature adapter, by applying a function $F : \mathbb{R}^N \to \mathbb{R}^M$, where F linearly maps the extracted features from the space $\mathbb{R}^N$ to a new feature space $\mathbb{R}^M$, effectively adapting them to the specific context. Another solution involves using the stop-gradient function ($sg[\cdot]$) [60] during the retraining of the backbone network.

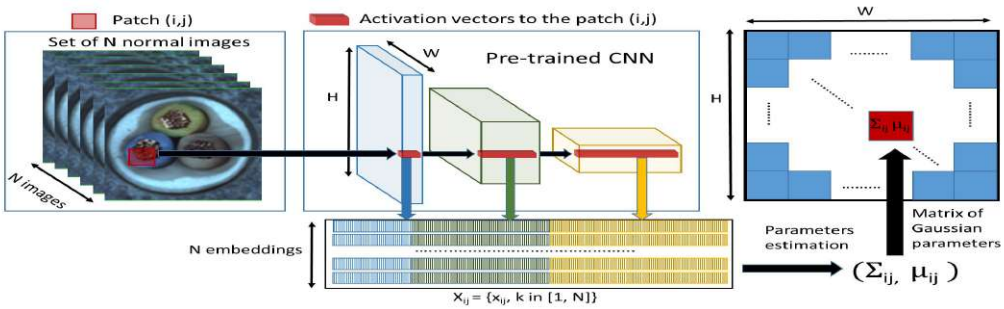
Moreover, embedding similarity-based models are often less explainable compared to generative models, as their logic operates at the level of deep embeddings, without providing an explicit reconstruction of the product. Finally, many state-of-the-art architectures rely on a so-called *Memory Bank* ($\mathcal{MB}$), an extensive set of features extracted during the training phase. In the case of large-scale datasets, this memory bank can require excessive amounts of RAM, making these solutions impractical for many real-world industrial applications.

Our approach, therefore, aims to tackle the latter issue by approximating the PatchCore algorithm. Although this approximation might lead to lower performance compared to the state-of-the-art results reported in [137], it ensures feasibility in contexts where hardware and time constraints are critical limiting factors.

8.3.2 PaDiM

To introduce the architecture of PaDiM [32], it is essential to note that this method, as an embedding similarity-based approach, leverages the capability of a pre-trained network on a large-scale sparse dataset, such as ImageNet, to aggregate and extract complex features from previously unseen images. The patch embedding process in PaDiM is analogous to that of SPADE [25], as illustrated in Figure 8.10.

During the training phase, each patch of the normal images is associated with its spatially corresponding activation vectors in the pretrained CNN activation maps. Activation vectors from different layers are then concatenated to



(a) PaDiM architecture

Figure 8.10: A scheme of the PaDiM architecture from [32].

form embedding vectors that encapsulate information from various semantic levels and resolutions, thereby encoding both fine-grained and global contexts. Given that the activation maps have a lower resolution than the input image, many pixels share the same embeddings, resulting in pixel patches that exhibit no overlap in the original image resolution. Consequently, an input image can be divided into a grid of positions $(i, j) \in [1, W] \times [1, H]$, where $W \times H$ represents the resolution of the largest activation map used to generate embeddings.

Ultimately, each patch position (i, j) in this grid is linked to an embedding vector x_{ij} computed as described above. Since the generated patch embedding vectors may carry redundant information, we investigate the potential for dimensionality reduction. The experimental results indicate that randomly selecting a subset of dimensions proves to be more efficient than employing a traditional PCA algorithm [109].

PaDiM operates by modeling each patch embedding as a multivariate Gaussian distribution, characterized by a mean vector μ_{ij} and a covariance matrix Σ_{ij} for each patch position (i, j). These parameters are estimated from the set of normal training patches. The covariance matrix Σ_{ij} captures the correlations between different semantic levels of the CNN, thus allowing PaDiM to effectively model the spatial distribution of normal features. This approach enables PaDiM to capture both fine-grained local structures and more global contextual information, improving anomaly detection accuracy.

At inference time, the Mahalanobis distance[2] is used to compare the em-

[2]for a detailed mathematical explanation of Mahalanobis distance and its implementa-

bedding of each test image patch, x_{ij}, to the learned Gaussian distribution $\mathcal{N}(\mu_{ij}, \Sigma_{ij})$. The resulting distance provides an anomaly score for each patch, and these scores are combined to form an anomaly map for the entire image. High scores in the anomaly map indicate regions that deviate significantly from the normal distribution, allowing for both detection and localization of anomalies.

One of PaDiM's key advantages is its computational efficiency at test time, as it does not require the use of k-NN algorithms, which scale poorly with larger datasets. Instead, PaDiM's time complexity is independent of the training dataset size, making it suitable for real-time industrial applications.

Finally, dimensionality reduction techniques such as random feature selection have proven effective in further optimizing PaDiM's performance. Experimental results show that reducing the number of dimensions in the patch embeddings to 100 or 200 can significantly decrease both training and inference time, while maintaining competitive performance compared to the full 448-dimensional embeddings. This balance between performance and computational efficiency positions PaDiM as a highly practical solution for anomaly detection and localization in various domains.

8.3.3 PatchCore

Another fundamental architecture that needs to be introduced to understand key concepts for the next sections is PatchCore [137]. PatchCore improves on previous methods by using a memory bank of locally aggregated patch-level features combined with efficient coreset subsampling, which allows for fast inference and accurate anomaly detection. This approach reduces computational overhead while maintaining state-of-the-art performance, making it suitable for real-time industrial applications.

To describe the architecture of PatchCore, depicted in Figure 8.11, it is important to note that this method, similar to PaDiM, leverages the capabilities of a pre-trained network on a large-scale dataset such as ImageNet. PatchCore extracts patch-level features from intermediate layers of a ResNet-based architecture and aggregates these features over local neighborhoods, capturing

tion, please refer to Appendix B.

both local variations and global contexts.

During the training phase, PatchCore creates a memory bank from the patch features of nominal (non-defective) images. Unlike methods that require extensive memory usage and computational resources, PatchCore applies a coreset subsampling algorithm to reduce the size of the memory bank. This algorithm selects the most representative patch features, ensuring that the memory bank remains small without sacrificing performance. The use of this coreset drastically reduces memory and computation overhead, making PatchCore particularly suitable for time-critical anomaly detection tasks.

Mathematically, let $\mathbf{x}_{ij}$ represent the embedding vector for a patch at position (i, j) in the input image. The memory bank is built from the embeddings of the nominal training images, denoted as $\mathcal{M} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, where N is the number of stored embeddings. To reduce the size of the memory bank, PatchCore uses a coreset reduction algorithm, which selects a subset $\mathcal{C} \subseteq \mathcal{M}$ that covers the nominal feature space efficiently, while minimizing the reconstruction error.

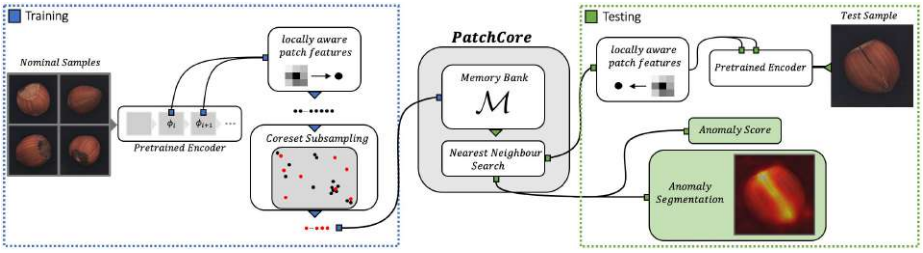
Formally, the coreset is obtained by solving a minimization problem over the set of training patches:

$$\mathcal{C} = \arg\min_{\mathcal{C}' \subseteq \mathcal{M}} \sum_{\mathbf{x} \in \mathcal{M}} \min_{\mathbf{c} \in \mathcal{C}'} \|\mathbf{x} - \mathbf{c}\|_2^2 \tag{8.17}$$

where $\|\cdot\|_2$ is the Euclidean distance, and $\mathcal{C}'$ represents a candidate coreset. This coreset subsampling technique ensures that PatchCore retains only the most relevant and representative embeddings, reducing both memory usage and inference time.

At inference time, PatchCore uses nearest-neighbor search to compare patch features from the test image against the coreset. Given an embedding $\mathbf{x}_{ij}$ from the test image, PatchCore computes the anomaly score based on the distance to the nearest neighbor in the coreset $\mathcal{C}$:

$$s_{ij} = \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x}_{ij} - \mathbf{c}\|_2 \tag{8.18}$$



(a) PatchCore architecture

Figure 8.11: A scheme of the PatchCore architecture from [137].

The anomaly scores s_{ij} are computed for each patch in the image, and the resulting scores are aggregated into an anomaly map. This anomaly map provides a detailed spatial representation of the regions in the test image that deviate from the nominal distribution.

One of the key innovations in PatchCore is the coreset subsampling, which enables the memory bank to be significantly reduced while retaining high accuracy in anomaly detection. This makes PatchCore both memory- and computation-efficient, while maintaining high detection performance. In experiments on the MVTec AD benchmark, PatchCore achieved state-of-the-art results with an AUROC score of 99.6%, more than halving the error compared to previous methods. PatchCore is also highly sample-efficient, achieving robust performance even when trained on a fraction of the available nominal data.

8.3.4 Approach Limitations

Although both approaches constitute excellent implementations for OOD-AD in industrial contexts, they have limitations that often prevent their application to large-scale datasets, such as those described previously.

PaDiM [32] computes the covariance matrix across the set of all extracted embeddings. This problem, due to the hardware limitation of RAM, can be addressed by using online covariance calculation algorithms and, more generally, by estimating the statistical parameters of the underlying multivariate Gaussian distribution [30], as thoroughly explained in Appendix C. Additionally, more optimized methods, such as using the Cholesky decomposition, can be employed as described in Appendix D. However, the idea of a hypersphere

centered on $\mu = \mathbb{E}[D]$, where $\mathbb{E}$ is the expected value and D is the training dataset, works well on MVTec example datasets where the defect is always clearly visible and detectable. But it proves less expressive in contexts where the defect is much more "in-distribution" at the feature level. In such cases, it becomes much more difficult to separate good examples from defective ones in the test dataset.

PatchCore [137], from this perspective, is much more capable of separating defective features from regular ones, at the cost of greater computational complexity, using a k-NN for distance calculations, and a much higher memory cost. Despite the aforementioned reduction of the memory bank via coreset subsampling, in cases where the dataset is extremely large, it may still be too costly to maintain even a subsample of the original memory bank.

For these two reasons, an alternative approach is being studied that aims to combine the strengths of both theories to overcome their limitations, albeit at the cost of reduced performance in terms of AUROC and accuracy in certain cases. However, this makes the approach feasible in real-world contexts.

To summarize the contribution of the proposed experiment, an embedding similarity-based architecture is tested, which can be applied in industrial contexts with large-scale datasets. The model, which is still under study, is being evaluated on several MVTec datasets for comparison with PatchCore and PaDiM.

8.3.5 Model Description

This work introduces a hybrid approach that integrates aspects of both the PaDiM and PatchCore architectures, targeting the reduction of dependency on extensive memory banks. This diminution is facilitated by reducing the memory footprint and leveraging statistical indices that more accurately characterize large-scale challenges, as opposed to the comprehensive k-NN search utilized by PatchCore [137]. Furthermore, the maintenance of centroids, as opposed to relying exclusively on the median of the embedding vectors, permits more precise threshold adjustments to distinguish between anomalous and regular features.

The operations can be summarized as follows:

- Initialization of the pre-trained backbone network.
- Extraction and aggregation of features from the training dataset.
- Optional reduction of features for the computation of the covariance matrix and clustering.
- Computation of covariance matrices and Cholesky decomposition matrices.
- Transformation of examples using the Cholesky decomposition.
- Computation of cluster centroids or reduction via the k-greedy coreset subsampling algorithm.
- Extraction and aggregation of features from the test dataset.
- Projection of features using the Cholesky decompositions.
- Calculation of k-NN using the Euclidean distance of the transformed features.
- Collection of metrics and statistics.

8.3.5.1 Backbone Network

In alignment with the methodologies described in [32, 137], the proposed model, designated as MhBC-PatchCore, incorporates a pretrained backbone network, specifically a *Wide ResNet50 v2*. The transformation function F, defined as

$$F : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H^* \times W^* \times C^*}, \tag{8.19}$$

extracts and concatenates features Φ from an input image X. The embedding vectors ϕ_{hw} are extracted at specified positions (h, w).

The initial step involves dimensionality reduction of the embedding vectors, concatenating three layers at varying depths to produce a pyramid-type feature extraction. Given that C^* is exceedingly large and challenging to handle with commercial hardware, two approaches are proposed for managing this complexity: PCA and Sparse Random Projection.

8.3.5.2 Feature Aggregation

The process of feature aggregation is performed using the aggregation function described in the work of Roth et al. [137].

Specifically, two deep layers are utilized to adopt a pyramidal approach, extracting two different levels of feature aggregation. Let

$$\phi_{i,j} = \phi_j(x_i) \in \mathbb{R}^{H_j \times W_j \times C_j} \tag{8.20}$$

be the feature tensor extracted at layer $j \in \{2, 3\} \subseteq \mathbb{N}$, where x_i is the input. Let

$$N_p^{(h,w)} = \{(a, b) \mid a \in [h - \lfloor p/2 \rfloor, \ldots, h + \lfloor p/2 \rfloor], b \in [w - \lfloor p/2 \rfloor, \ldots, w + \lfloor p/2 \rfloor]\} \tag{8.21}$$

be the neighborhood function that incorporates the features from the patch of size p.

As described in the reference work:

$$\phi_{i,j}^{N_p(h,w)} = f_{\text{agg}}\left(\{\phi_{i,j}(a, b) \mid (a, b) \in N_p^{(h,w)}\}\right), \tag{8.22}$$

where f_{agg} is an aggregation function of the feature vectors in the neighborhood $N_p^{(h,w)}$. For the described architecture, we use adaptive average pooling on the features extracted by each layer after patch extraction. The deeper layer is then upsampled, using H_0 and W_0 as reference dimensions, followed by the concatenation of features C_j, where $j \in \{0, 1\}$. Finally, adaptive average pooling is applied to the concatenated features, defining $\Phi(x_i)$ as the final tensor of features extracted from sample x_i.

8.3.5.3 Features Reduction

The reduction of extracted features serves various purposes. Although feature reduction inherently removes information, this information is often redundant or highly correlated. As will be discussed, reducing the collected features is often necessary to ensure numerical stability in the covariance matrix. Furthermore, clustering algorithms, which are often used to reduce the dimensionality

of the problem at the sample level, suffer from the so-called curse of dimensionality. This issue generally arises when the number of data points is small (in a suitably defined sense) relative to the intrinsic dimensionality of the data. To address this, three different dimensionality reduction methods have been proposed. The last method is the one implemented in the official PatchCore and is used exclusively during coreset reduction via a greedy algorithm. Other calculations, such as k-NN, are performed on the features extracted through $\Phi = f_{\text{agg}}$.

8.3.5.3.1 Principal Component Analysis (PCA) PCA is a statistical technique employed to reduce the dimensionality of a dataset while preserving as much variance as possible. It is particularly useful in processing high-dimensional data such as image features extracted from deep learning models. In the context of our proposed architecture, PCA is applied to the high-dimensional tensor Φ, which represents the concatenated features from various depths of the network, aiming to produce a more compact yet informative representation.

8.3.5.3.1.1 Reshaping and Flattening the Data Initially, the tensor $\Phi \in \mathbb{R}^{H \times W \times C}$ is reshaped into a matrix form to facilitate mathematical operations. This is done by flattening the spatial dimensions H (height) and W (width), resulting in a new matrix $X \in \mathbb{R}^{N \times C}$, where $N = H \times W$ is the total number of spatial locations, and C is the number of channels:

$$X = \begin{bmatrix} \phi_{11} \\ \phi_{12} \\ \vdots \\ \phi_{HW} \end{bmatrix} \tag{8.23}$$

8.3.5.3.1.2 Centering the Data To apply PCA, the data must be centered around the mean. The mean vector $\mu \in \mathbb{R}^C$ is calculated by averaging

all feature vectors across the spatial locations:

$$\mu = \frac{1}{N} \sum_{n=1}^{N} X_n \tag{8.24}$$

Each feature vector in X is then centered by subtracting μ from every row, resulting in a matrix of centered data X_{centered}:

$$X_{\text{centered}} = X - \mu \tag{8.25}$$

8.3.5.3.1.3 Computing the Covariance Matrix The covariance matrix $C \in \mathbb{R}^{C \times C}$ is computed from the centered data. This matrix captures the variance and covariance between every pair of features across the dataset:

$$C = \frac{1}{N} X_{\text{centered}}^{\top} X_{\text{centered}} \tag{8.26}$$

8.3.5.3.1.4 Eigenvalue Decomposition The next step involves decomposing the covariance matrix C to extract its eigenvectors and eigenvalues. This decomposition is crucial as it reveals the principal components of the data:

$$C = V \Lambda V^{\top} \tag{8.27}$$

Here, $V \in \mathbb{R}^{C \times C}$ contains all eigenvectors (principal components), and $\Lambda \in \mathbb{R}^{C \times C}$ is a diagonal matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_C$, sorted in descending order. These eigenvalues represent the variance captured by their corresponding eigenvectors.

8.3.5.3.1.5 Selecting Principal Components To achieve dimensionality reduction, the top C' eigenvectors, corresponding to the largest C' eigenvalues, are selected. These components capture the most significant variance

within the data. The selection of C' depends on the desired level of data compression and the variance one wishes to retain:

$$V_{C'} = \text{first } C' \text{ columns of } V \tag{8.28}$$

8.3.5.3.1.6 Projecting the Data The original high-dimensional data is then projected onto the space spanned by the selected eigenvectors, reducing its dimensions:

$$X' = X_{\text{centered}} V_{C'} \tag{8.29}$$

8.3.5.3.1.7 Reshaping Back to Tensor Format Finally, the reduced data $X' \in \mathbb{R}^{N \times C'}$ is reshaped back into tensor format, matching the original spatial dimensions but with reduced channel depth:

$$\Phi' = \text{reshape}(X', [H, W, C']) \tag{8.30}$$

This transformed tensor Φ' now serves as a reduced representation of the original features, ready for further processing or analysis within the model pipeline.

8.3.5.3.2 Sparse Random Projection (SRP) Sparse Random Projection (SRP) is a dimensionality reduction technique based on projecting the original data onto a sparse random matrix. This method offers computational efficiency and works well in preserving distances between high-dimensional data points.

8.3.5.3.2.1 Random Matrix Generation The projection matrix R used in SRP is sparse and randomly generated as follows:

$$R_{ij} = \begin{cases} \sqrt{\frac{s}{d}} & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \\ -\sqrt{\frac{s}{d}} & \text{with probability } \frac{1}{2s} \end{cases} \tag{8.31}$$

Here, s controls the sparsity of the matrix, and d is the original dimensionality. The sparse nature of R reduces the computational cost compared to dense matrices, making it well-suited for large-scale data.

8.3.5.3.2.2 Projecting Data Given the sparse random matrix $R \in \mathbb{R}^{C \times C'}$, the data tensor $\Phi \in \mathbb{R}^{H \times W \times C}$ is projected onto a lower-dimensional space:

$$\Phi' = \Phi R \tag{8.32}$$

This projection approximates the pairwise distances between the original data points, ensuring that essential geometric properties of the data are preserved. Sparse random projections are particularly effective for reducing dimensionality when dealing with very high-dimensional data, such as features extracted from neural networks.

8.3.5.3.2.3 Computational Efficiency and Theoretical Guarantees One of the primary advantages of SRP is its efficiency. The sparse structure of R significantly reduces both memory requirements and computational load during matrix multiplication. Additionally, SRP relies on the Johnson-Lindenstrauss lemma, which guarantees that the pairwise distances between data points are approximately preserved after projection. Mathematically, the Johnson-Lindenstrauss lemma states that for any two points x and y, after projection, their Euclidean distance will satisfy:

$$(1 - \epsilon)\|x - y\|^2 \leq \|Rx - Ry\|^2 \leq (1 + \epsilon)\|x - y\|^2 \tag{8.33}$$

where ϵ is a small distortion factor. This ensures that the geometric structure of the data is preserved during dimensionality reduction, making SRP a robust method for tasks such as feature compression and anomaly detection.

8.3.5.3.3 Random Projection in PatchCore Implementation In the official PatchCore implementation by Amazon©[3], a randomly initialized Linear layer is instantiated and not trained, serving to reduce the feature dimensionality. This is achieved by projecting the original feature tensor ϕ onto a lower-dimensional space using a randomly initialized weight matrix W:

$$\Phi' = \Phi W \tag{8.34}$$

where W is the weight matrix randomly instantiated at the beginning of the process. This random projection serves the purpose of reducing the dimensionality of the feature space, ensuring that the computational cost remains low while still preserving key information from the original features. The weights in W remain fixed throughout the training and inference stages, reflecting a non-parametric dimensionality reduction approach within PatchCore.

This technique is only used to address the curse of dimensionality during coreset reduction.

8.3.5.4 Covariance Tensor and Cholesky Decomposition

For each feature map Φ_i extracted from the input x_i, a covariance tensor Σ is computed. The tensor $\Sigma \in \mathbb{R}^{H_0 \times W_0 \times C_{\text{agg}} \times C_{\text{agg}}}$ represents, at each spatial location (h, w), a covariance matrix $\Sigma^{(h,w)}$. This covariance matrix $\Sigma^{(h,w)}$ captures the multivariate stochastic relationships among the embedding vectors ${\phi_i}^{(h,w)}$, which describe the features of the input data at that particular location.

The computation of the covariance tensor is critical in scenarios where the dimensionality of the embedding space is high, and the relationships among features need to be modeled accurately. However, this process can be computationally expensive and memory-intensive, especially when dealing with large-

[3] `https://github.com/amazon-research/patchcore-inspection` by Amazon Research

scale datasets. To mitigate these issues, the **Online Covariance Calculation** method is employed. As detailed in Appendix C, this method incrementally updates the covariance matrix by processing the data in batches. This allows the model to efficiently handle data streams and extensive datasets by significantly reducing the memory footprint while maintaining the statistical properties of the covariance matrices across the dataset.

Once the covariance matrices $\Sigma^{(h,w)}$ are computed at each spatial location, the next step involves performing the Cholesky decomposition. The Cholesky decomposition is a factorization method that decomposes a positive definite matrix into the product of a lower triangular matrix and its transpose. In this case, for each covariance matrix $\Sigma^{(h,w)}$, a corresponding lower triangular matrix $L^{(h,w)}$ is calculated:

$$\Sigma^{(h,w)} = L^{(h,w)} {L^{(h,w)}}^{\top} \tag{8.35}$$

The Cholesky decomposition is beneficial because it provides a numerically stable way of transforming the covariance matrix into a simpler form, which can then be used to project the original embedding vectors $\phi_i^{(h,w)}$ into a new space. The projection of each embedding vector is defined as follows:

$$z^{(h,w)} = {L^{(h,w)}}^{-1} \phi_i^{(h,w)} \tag{8.36}$$

This projection $z^{(h,w)}$ transforms the original embeddings $\phi_i^{(h,w)}$ into a new space where the multivariate properties of the data are better aligned for anomaly detection or other subsequent tasks. However, explicitly computing the inverse ${L^{(h,w)}}^{-1}$ can introduce numerical instability and additional computational overhead, especially for large matrices.

To address this issue, rather than explicitly calculating ${L^{(h,w)}}^{-1}$, the implementation leverages the `triangular_solve` operation. This operation efficiently solves the system of linear equations defined by the triangular matrix $L^{(h,w)}$ and the embedding vector $\phi_i^{(h,w)}$ without needing to invert $L^{(h,w)}$. The

system of equations is:

$$L^{(h,w)} z^{(h,w)} = \phi_i^{(h,w)} \tag{8.37}$$

By solving this system directly, the `triangular_solve` operation avoids the instability and computational cost associated with matrix inversion, while still providing the desired transformation $z^{(h,w)}$. Since $L^{(h,w)}$ is a lower triangular matrix, this method takes advantage of its structure, improving both numerical stability and computational efficiency. The use of `triangular_solve` thus ensures that the projection $z^{(h,w)}$ can be computed reliably, even for high-dimensional data, without sacrificing performance or precision.

8.3.5.5 Clustering

8.3.5.5.1 K-Means Coreset Subsampling Once Ψ, the tensor of embedding vectors Φ projected via Cholesky decomposition metrics, has been obtained, clustering is performed using a cumulative K-Means algorithm. This process aims to partition the embedding vectors into K clusters, each represented by a centroid. For each spatial position (h, w), the K-Means algorithm minimizes the within-cluster sum of squared distances between each point and its assigned centroid. Mathematically, given a set of N points, $\{\psi_i\}_{i=1}^{N} \subset \mathbb{R}^d$, the K-Means optimization problem is defined as:

$$\min_{C_1, C_2, \ldots, C_k} \sum_{i=1}^{N} \min_{j=1,\ldots,k} \|\psi_i - C_j\|^2 \tag{8.38}$$

where: - $C_j \in \mathbb{R}^d$ is the centroid of cluster j, - $\|\psi_i - C_j\|^2$ represents the squared Euclidean distance between point ψ_i and the centroid C_j.

This iterative process alternates between assigning each point ψ_i to the nearest centroid C_j and updating each centroid C_j as the mean of the points assigned to it. The result is a set of k centroids that represent the clusters, leading to a coreset reduction from N to K, where N is the size of the initial dataset, and K is the number of centroids, typically with $N \gg K$.

8.3.5.5.2 K-Greedy Coreset Subsampling

PatchCore utilizes a coreset reduction method based on a greedy algorithm, which selects points incrementally to minimize the overall reconstruction error. This algorithm works by selecting points that cover the dataset in terms of distance, ensuring that each selected point is a good representative of a region of the dataset. Mathematically, at each step, the algorithm selects the point p_i from the dataset D that maximizes the minimum distance to any previously selected points S:

$$p_i = \arg\max_{p \in D \setminus S} \min_{q \in S} d(p, q) \tag{8.39}$$

where $d(p, q)$ represents the distance between points p and q. However, this greedy algorithm cannot be computed in an exact cumulative manner, as it relies on the relative distances between all points in the dataset. Even cumulative approximations would lead to significant inaccuracies due to the algorithm's dependency on the global structure of the data. As a result, the need to keep the entire dataset in memory renders this method impractical when dealing with large-scale datasets and/or high-dimensional feature vectors.

8.3.5.6 Distance Calculation

Finally, as in PatchCore [137], the respective distances are computed using k-NN with $k = 1$, based on the features extracted and projected through the Cholesky decompositions calculated in Section 8.3.5.4. The k-NN is performed using the Euclidean distance, defined as:

$$d(\psi_i, \psi_j) = \|\psi_i - \psi_j\|_2 = \sqrt{\sum_{c=1}^{C'} (\psi_i(c) - \psi_j(c))^2} \tag{8.40}$$

where ψ_i and ψ_j are the feature vectors projected via Cholesky decomposition, and C' is the reduced dimensionality of the feature space. In the original feature space, this corresponds to the Mahalanobis distance, as discussed in Appendix D.

This procedure allows for the training of a threshold that best separates

normal (good) samples from defective ones, according to the selected evaluation metric. For consistency with the original papers, the AUROC has been chosen as the primary evaluation metric. However, other metrics have also been calculated and reported for completeness, as AUROC is not always the most indicative measure, particularly in cases of highly imbalanced datasets.

8.3.5.7 Experiment

The study is ongoing, and preliminary experiments have been conducted on the benchmark MVTec datasets for performing AD (Anomaly Detection). In many cases, the performance of the proposed architecture exceeds that of the current state-of-the-art reference, PatchCore, which itself had previously surpassed its predecessor, PaDiM. Furthermore, the proposed architecture proves to be more feasible for large-scale industrial datasets, as it imposes fewer memory constraints compared to PatchCore. Specifically, PatchCore requires the complete extraction of features from the entire dataset for subsampling, which amounts to storing $N \times H \times W \times C$ features, where N is the number of images in the training set, and $H \times W \times C$ are the spatial and channel dimensions of the feature maps.

This large memory requirement in the training phase is a key limitation of PatchCore. Although PatchCore eventually reduces the features to a smaller core-set of size $K \times H \times W \times C$, this reduction occurs only after the full feature extraction, which necessitates considerable memory in the early stages of training. In contrast, PaDiM, while also having significant memory usage due to the need to store both mean and covariance matrices, does not require this initial full feature storage.

This advantage of the proposed architecture comes at the cost of increased computational complexity, as it requires the calculation of $H_0 \times W_0$ covariance matrices, their respective Cholesky decompositions, and the projection of the extracted embedding vectors. Despite the additional computation, the memory efficiency makes this approach suitable for industrial-scale datasets.

Table 8.1 presents a detailed comparison of the training complexity, testing complexity, and memory occupancy for PatchCore, PaDiM, and the proposed MhBC-PatchCore model. Each of these models has different characteristics in terms of computational complexity and memory efficiency, which are critical

considerations in real-world applications where resources are limited.

8.3.5.7.1 Explanation by Symbols The symbols used in Table 8.1 are explained as follows:

- N: The number of images in the training dataset.
- M: The number of images in the testing dataset.
- P: The number of patches extracted from each image.
- K: The size of the core-set after the reduction.
- $H \times W \times C$: The spatial dimensions ($H \times W$) and the number of channels (C) of the extracted feature maps.
- B: The computational complexity associated with the backbone network used for feature extraction.
- $\log(P')$: This factor appears in the nearest neighbor search process, as PatchCore utilizes approximate k-NN algorithms to reduce the complexity of the search over the core-set patches.
- λ: The memory required for storing auxiliary information, such as the core-set indices, nearest neighbor indices, and any pre-computed matrices necessary for inference.
- L^{-1}: In the case of MhBC-PatchCore, the computation of $z = L^{-1}x$, where L^{-1} is the inverse of a pre-computed matrix, adds to the overall testing complexity, along with the computation of k-NN as in PatchCore. However, since L^{-1} is already computed and stored, its contribution to memory is not significant.

8.3.5.7.2 Explanation by Model

- **PatchCore**: The training complexity for PatchCore involves extracting features for each patch in the training images, followed by core-set reduction. Initially, PatchCore requires the storage of all extracted features, which amounts to $N \times H \times W \times C$, before reducing them to $K \times H \times W \times C$.

This leads to significant memory consumption during the training phase. The testing complexity primarily consists of applying the backbone network to each test image and performing a nearest neighbor search over the reduced core-set of patches.

- **PaDiM**: In PaDiM, the training complexity is dominated by the computation of the covariance matrices for each patch, which scales quadratically with the feature dimensionality. PaDiM's memory usage is significant due to the storage of both the mean and covariance matrices for each patch. However, unlike PatchCore, PaDiM does not require full feature storage at the beginning of training.

- **MhBC-PatchCore**: The training complexity for MhBC-PatchCore is the sum of the complexities of PatchCore and PaDiM, as it integrates the core-set reduction approach of PatchCore with the Gaussian modeling of PaDiM. The testing complexity is similar to PatchCore but with an additional term $O(M \cdot D^2)$ for calculating $z = L^{-1}x$, where L^{-1} is already available. The memory occupancy is equivalent to PaDiM, as it requires storing the mean and covariance matrices for each patch.

Model	**Train**	**Test**	**Memory**
PatchCore	$O(N \cdot P \cdot B + P' \cdot P \cdot D)$	$O(M \cdot P \cdot B + M \cdot P \cdot P' \log P')$	$\lambda + N \cdot P \cdot D$ (before reduction), $\lambda + K \cdot P \cdot D$ (after reduction)
PaDiM	$O(N \cdot P \cdot (D + D^2))$	$O(M \cdot P \cdot (D + D^2))$	$\lambda + P' \cdot (D + D^2)$
MhBC	$O(N \cdot P \cdot B + P' \cdot P \cdot D + N \cdot P \cdot (D + D^2))$	$O(M \cdot P \cdot B + M \cdot P \cdot P' \log P' + M \cdot D^2)$	$\lambda + P' \cdot (D + D^2)$

Table 8.1: Training, testing complexity, and memory occupancy for PatchCore, PaDiM, and MhBC-PatchCore.

8.3.5.7.3 Results Table 8.2 presents the comparative performance of PatchCore and the proposed MhBC-PatchCore model across multiple datasets from the MVTec benchmark. The metrics used for evaluation include Accuracy, F1 score, Precision, Recall, and AUROC. These metrics provide a comprehensive overview of how each model performs in terms of both classification accuracy and anomaly detection capabilities.

For each dataset, the performance of both models is reported. Notably, MhBC-PatchCore demonstrates improvements in several key areas, particularly in Precision and AUROC, which are crucial for detecting anomalies with minimal false positives. The overall mean values, presented at the bottom of the table, highlight the marginal performance gains achieved by MhBC-PatchCore over the standard PatchCore. Specifically, the MhBC-PatchCore model outperforms PatchCore in terms of mean Accuracy, F1, Precision, and AUROC, confirming the effectiveness of the modifications introduced in this architecture.

These results suggest that the enhancements in the MhBC-PatchCore model provide a more robust solution for anomaly detection, especially in scenarios where high precision and recall are required. Furthermore, the consistent improvement in AUROC indicates better performance in ranking anomalies, a critical factor for industrial applications.

Dataset	**PatchCore**					**MhBC-PatchCore**				
	Accuracy	**F1**	**Precision**	**Recall**	**AUROC**	**Accuracy**	**F1**	**Precision**	**Recall**	**AUROC**
bottle	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
carpet	0.957	0.971	1.000	0.944	0.981	0.957	0.971	1.000	0.944	0.976
cable	0.980	0.983	1.000	0.967	0.992	0.973	0.978	0.978	0.978	0.993
capsule	0.924	0.956	0.923	0.991	0.947	0.955	0.972	0.972	0.972	0.954
grid	0.897	0.934	0.877	1.000	0.945	0.859	0.903	0.911	0.895	0.921
hazelnut	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
leather	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
metal nut	0.974	0.984	0.989	0.978	0.991	0.983	0.989	0.989	0.989	0.997
pill	0.898	0.940	0.937	0.943	0.914	0.928	0.957	0.957	0.957	0.944
screw	0.863	0.913	0.865	0.966	0.887	0.844	0.901	0.851	0.958	0.897
tile	0.991	0.994	1.000	0.988	0.999	0.991	0.994	0.988	1.000	1.000
toothbrush	0.881	0.923	0.857	1.000	0.897	0.905	0.938	0.882	1.000	0.894
transistor	0.960	0.951	0.929	0.975	0.994	0.990	0.987	1.000	0.975	0.999
wood	0.975	0.983	1.000	0.967	0.994	0.987	0.992	1.000	0.983	0.997
zipper	0.940	0.963	0.937	0.992	0.971	0.967	0.979	0.960	1.000	0.952
MEAN	0.949	0.966	0.954	**0.981**	0.967	**0.956**	**0.971**	**0.966**	0.977	**0.968**

Table 8.2: Comparison of PatchCore and MhBC-PatchCore Performances

8.3.5.8 Approach Limitations

The primary limitations of approaches based on embedding similarity lie in the fact that they do not involve training a neural model. These approaches rely on the premise, as discussed in Section 8.3.1, that the models are pretrained on large, diverse datasets, such as ImageNet, and are thus capable of aggregating complex features, including those that may not have been seen during training. In general, this assumption has been experimentally validated,

(a) A floating particle

Figure 8.12: A floating particle stuck on the internal surface of the product.

especially in datasets where the data is well represented, for example, in images with well-defined contours, colors, and resolution, where the OOD anomaly is markedly distinct from the background, as is often the case with benchmark datasets.

However, in more real-world scenarios, this is not always the case. For instance, in Figure 8.12, a particle trapped on the inner surface of a vial during the BFS process does not exhibit a sharp gradient in terms of grayscale contrast relative to its surroundings, and it occupies a very limited area. In such cases, the embedding similarity approaches struggle to detect subtle defects that are not prominently distinguished from the background.

Moreover, these approaches are constrained by the use of pre-trained networks, which typically have fixed input dimensions and a pre-defined number of channels. This limitation can make them less flexible and not always applicable to the current problem, especially when the input characteristics do not align with the pre-trained model's expected configuration.

8.3.5.9 Ablation Study

Two ablation experiments were conducted to evaluate the impact of specific components of the proposed approach:

1. The use of a single covariance matrix for all feature arrays.

2. The use of PatchCore with Euclidean distance combined with online K-

Means for coreset subsampling.

The results demonstrated that the initial assumptions and architectural choices led to better performance compared to the ablations presented. Specifically, the proposed approach, which includes multiple covariance matrices and the tailored coreset subsampling strategy, outperformed the simplified versions tested in the ablation study.

As shown in Table 8.3, the results of the ablation studies, particularly the *Cumulative K-Means*, were significantly lower in terms of AUROC compared to the reference MhBC-PatchCore, reported in Table 8.2.

AUROC	Cumulative K-Means	Single Covariance Matrix	MhBC-PatchCore
bottle	0.998	1.000	1.000
hazelnut	0.964	1.000	1.000
carpet	0.923	0.962	0.976
cable	0.787	0.908	0.993
capsule	0.671	0.962	0.954
grid	0.509	0.609	0.921
leather	0.991	1.000	1.000
metal Nut	0.791	0.994	0.997
pill	0.671	0.939	0.944
screw	0.474	0.788	0.897
tile	0.981	0.956	1.000
toothbrush	0.767	0.892	0.894
transistor	0.684	0.927	0.999
wood	0.968	0.979	0.997
zipper	0.926	0.967	0.952

Table 8.3: AUROC comparison between the two ablation studies.

8.3.6 Related Work

Research in anomaly detection (AD) has spanned several methodologies, which can be broadly classified into three categories: reconstruction-based methods, embedding similarity-based methods, and image synthesis generation-based methods. These categories often overlap in their implementations and can complement one another depending on the specific application domain.

Reconstruction-based approaches focus on training models to learn normal patterns by reconstructing the input data and then identifying anomalies based on reconstruction error. These methods are discussed in detail in Section 8.2.4, where AEs, VAEs, GANs, and Diffusion Models are explored for their use in anomaly detection.

A further prominent methodology within AD is the *embedding similarity-based* approach. This strategy involves extracting feature vectors, or embeddings, from entire images or image patches for the purpose of detecting anomalies [134, 12, 116]. An anomaly score is computed by measuring the distance between the embedding vectors of a test image and those representing normal images, which were extracted during training or initialization. Notable methods in this category include SPADE [26], PaDiM [32], and PatchCore [137], each optimizing the usage of embeddings to ensure lower inference costs while maintaining performance. Despite their promising results, embedding similarity-based approaches often lack interpretability, as they do not directly indicate the specific features contributing to the anomaly score. Additionally, these methods typically rely on models pre-trained on large-scale datasets like ImageNet [139], and their performance can suffer when dealing with more subtle or context-specific anomalies, such as those encountered in real-world industrial settings.

Within the realm of embedding similarity, generative models known as *normalizing flows* have gained prominence. These models, such as NFLOW, are particularly adept at estimating likelihoods for OOD examples by transforming complex distributions into simple, tractable ones [35]. In the context of AD, normalizing flows have been used to model the distribution of normal data and detect anomalies by identifying points that fall outside this learned distribution. Notable implementations include DifferNet [138], CFLOW-AD [57], and PNI [6], which incorporate position and neighborhood information within the distribution of normal features, further refining the anomaly detection process.

Knowledge distillation techniques have also gained significant traction in AD, particularly in handling large images with limited computational resources. These methods distill knowledge from a large, complex model into a smaller, more efficient model, allowing for faster inference without sacrificing accuracy. This is especially important for real-time applications. Works such as *Beyond Dents and Scratches* by Bergmann et al. [13] and EfficientAD by Batzner et al. [9] have shown the effectiveness of distillation techniques in reducing processing time while maintaining high performance in unsupervised ML settings. These techniques are increasingly used in industrial applications where fast, reliable anomaly detection is crucial.

Image synthesis generation-based methods focus on generating new samples that resemble normal data distributions, which are then used as a reference for detecting anomalies. This approach often involves the use of generative models, such as GANs [56] or Diffusion Models [5, 151, 81, 72, 73, 118], to synthesize images that closely mimic the normal data distribution. A key motivation behind image synthesis is to augment the training dataset by artificially creating OOD samples (i.e., anomalies). This helps to increase the frequency and diversity of anomalies, which are often scarce in real-world datasets. Methods such as DRÆM [170] and GRD-Net [43] fall under this category. Both models combine aspects of reconstruction-based methods and image synthesis, aiming to generate plausible defective examples, thus improving the model's ability to detect anomalies. These synthesized anomalies are then used to train the model to distinguish between normal and abnormal features more effectively. In addition, DRÆM utilizes a reconstruction-discrimination framework where synthetic defects are added to normal images to simulate anomalies. The network learns to reconstruct normal parts of the image while focusing on distinguishing defects, making it a hybrid approach that blends image synthesis with reconstruction-based techniques. Similarly, GRD-Net integrates generative and discriminative models, combining synthesis with anomaly detection to enhance performance, particularly in industrial applications where defects are rare and diverse. Overall, image synthesis generation-based methods not only augment the data available for training but also improve the detection of subtle and rare anomalies, especially when used in conjunction with other approaches like embedding similarity or reconstruction-based methods.

Part IV

Industrial Applications and Integration of Anomaly Detection Algorithms

Chapter 9

Experimental Results on GRD-Net

A series of experiments were carried out to evaluate the performance of GRD-Net. The model was compared with DRÆM and GANomaly, two state-of-the-art reconstruction-based anomaly detection and localization techniques. The architecture of GRD-Net, featuring a residual convolutional autoencoder, includes two residual blocks, as illustrated in Figure 8.2.

The experiments involved training both vanilla DRÆM AE and the GAN with residual AE for 200 epochs. The network design draws inspiration from the ResNet V2 architecture, which is commonly employed for classification tasks [65].

GRD-Net [43] was evaluated on both a subset of benchmark datasets from MVTec and a large pharmaceutical dataset, composed of strips of 5 vials in BFS. The findings demonstrated that the proposed architecture consistently outperformed conventional generative networks based on convolutional AEs. Summarizing the contents of this section:

1. Evaluation of GRD-Net in comparison with DRÆM and GANomaly for anomaly detection and localization.

2. The architecture and design of GRD-Net, incorporating residual blocks within a convolutional autoencoder.

3. Performance analysis of GRD-Net on benchmark datasets from MVTec and a proprietary pharmaceutical dataset.

4. Training protocol for the generative and discriminative models, including the impact of data augmentation and learning rate strategies.

5. Comparative results highlighting GRD-Net's effectiveness on diverse datasets, including a real-case pharmaceutical application.

The first part of the experiments was conducted using four challenging datasets from MVTec's sets: hazelnut, metal nut, pill, and cable datasets. In the second part, the network was tested on hazelnut, zip, and a proprietary pharmaceutical set of BFS strips of vials from a real study and use case conducted at Bonfiglioli Engineering, for a quality control VI machine. For these datasets, a second ROI dataset was prepared for each training nominal image.

9.1 MVTec AD Benchmark Datasets

As mentioned above, the model was tested on four different MVTec datasets. The decision to limit the testing to only four datasets was primarily due to the industrial focus and the goal of demonstrating performance on a realistic case. These reduced datasets do not offer a valid comparison for large-scale datasets where products are very similar across images in terms of position, edges, and color. In such cases, approaches based on *embedding similarity* tend to be less expressive, as described in Section 8.3.1, and are less capable of detecting anomalies in this specific context, especially in the presence of small details. This is further corroborated by the implementation-related considerations discussed in Section 8.3.4.

9.1.1 GAN with Residual AE

As mentioned in the summary, the first experiment aims to challenge the vanilla version of the DRÆM architecture. The training lasted 200 epochs, instead of the 700 used for testing DRÆM in the original paper, to provide a more realistic case, which can be implemented on a production line in a real industrial field. During this step, anomaly detection per image and defect localization within the image were evaluated. The learning rate was set to 10^{-4}, and a policy based on the "*reduction on plateau*" heuristic was used, with a patience of

three epochs and a reduction factor $\alpha = 0.1$. When a plateau of three epochs was reached at epoch k, the learning rate decreased using the formula:

$$\mathcal{LR}_k = \mathcal{LR}_{k-1} \cdot e^{-\alpha}. \tag{9.1}$$

where $\mathcal{LR}_k$ is the learning rate at the k-th epoch.

For the evaluation, AUROC, which is widely used in architecture comparisons, was applied at both image-level and pixel-level, as semi-supervised AD and AnoSeg scores.

Data augmentation was performed on training examples using random rotation in the range of $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ radians, to reduce overfitting during training over many epochs, due to the small number of anomaly-free images provided in the MVTec datasets.

For the sake of completeness, GRD-Net was also tested and compared with a vanilla convolutional AE (without residual blocks), and GRD-Net with fully-convolutional ResAE.

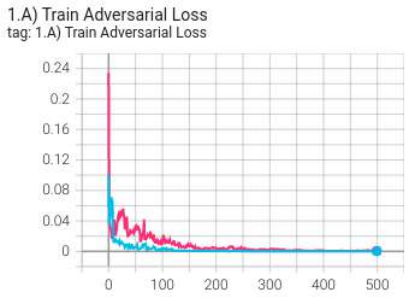
The experiment was performed using a large-scale pharmaceutical dataset over 500 epochs, using only the generative part, and comparing the losses. This is because the second part depends strictly on the performance of the first.

The experimental results are very encouraging and support the intuition that the residual network, even in the case of an AE applied to a GAN, is more effective in generating the final data.

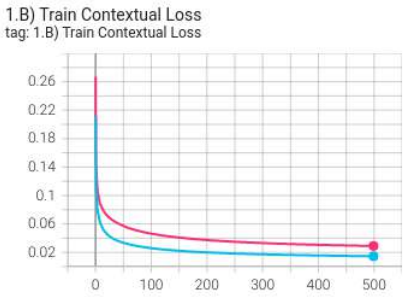
	Loss value for training (validation) phase			
	Epoch 250 vanilla	Epoch 250 residual	Epoch 500 vanilla	Epoch 500 residual
Adversarial Loss	2.8811×10^{-3} (5.1180×10^{-2})	3.2498×10^{-4} (3.8928×10^{-3})	7.4750×10^{-4} (1.3720×10^{-3})	1.3797×10^{-4} (4.5011×10^{-4})
Contextual loss	0.03502 (0.04136)	0.01853 (0.02782)	0.02912 (0.03747)	0.01460 (0.02488)
Encoder Loss	1.7221×10^{-4} (2.1078×10^{-4})	6.8905×10^{-5} (1.2333×10^{-4})	1.1745×10^{-4} (1.6965×10^{-4})	4.6982×10^{-5} (1.4363×10^{-4})
SSIM Loss	0.02665 (0.03115)	0.01304 (0.02010)	0.02200 (0.02830)	0.01014 (0.01872)

Table 9.1: Comparison between *vanilla* and *residual* architectures used for the generative part of the GAN architecture in GRD-Net.

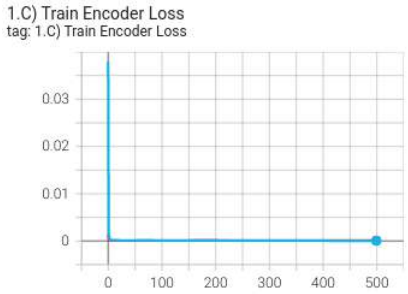
This can be visually appreciated in Figure 9.1. A comparison between



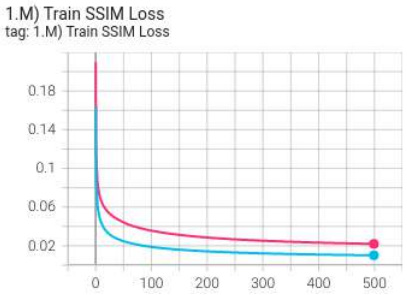
(a) Training adversarial loss (magenta for vanilla and cyan for residual)



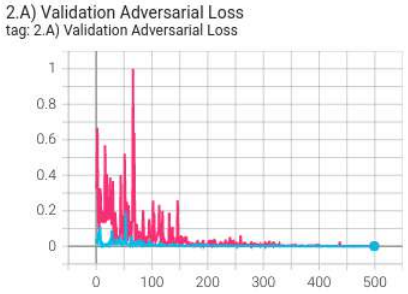
(b) Training contextual loss (magenta for vanilla and cyan for residual)



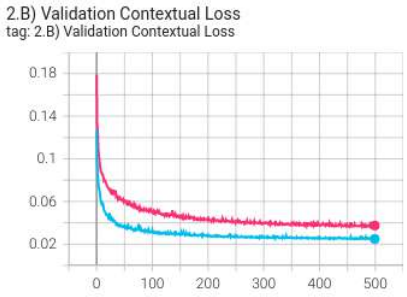
(c) Training encoder loss (magenta for vanilla and cyan for residual)



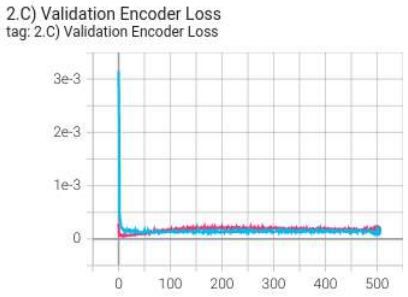
(d) Training SSIM loss (magenta for vanilla and cyan for residual)



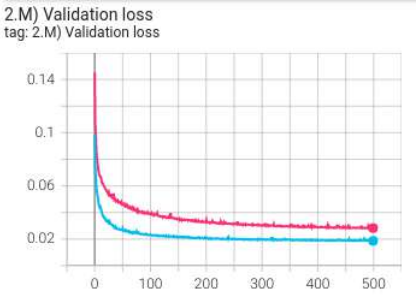
(e) Validation adversarial loss (magenta for vanilla and cyan for residual)



(f) Validation contextual loss (magenta for vanilla and cyan for residual)



(g) Validation encoder loss (magenta for vanilla and cyan for residual)



(h) Validation SSIM loss (magenta for vanilla and cyan for residual)

Figure 9.1: Visual representation of how the network with vanilla AE (magenta) is not only less effective but also noisier in some losses, such as adversarial loss, compared to the residual architecture (cyan).

losses used for the generator is also provided in Table 9.1.

9.1.1.1 Anomaly Detection

As illustrated in Table 9.2, GRD-Net demonstrates comparable and, in certain instances, superior performance to both DRÆM and PatchCore. During training, data augmentation was employed with a transformation function:

$$\overset{\star}{x}_i = t_p^{\theta}(x_i), \tag{9.2}$$

where p represents the probability of applying the transformation, and θ is a set of parameters defining the transformation. The transformed $\overset{\star}{x}_i$ is still considered part of the regular dataset. This augmentation was essential due to the limited size of the original datasets, which were insufficient to effectively train models such as AEs or GANs, including GRD-Net.

	AUROC per image (pixel)				
	GANomaly	DRÆM	PaDiM	PatchCore	GRD-Net
hazelnut	78.5	100.0	-	100.0	100.0
	(-)	(95.0)	97.7	98.6	(97.4)
metal nut	70.0	98.7	-	99.7	100.0
	(-)	(86.7)	96.7	98.4	(96.2)
pill	74.3	97.9	-	97.0	98.5
	(-)	(94.8)	94.7	97.1	(95.8)
cable	75.7	91.8	-	99.3	99.5
	(-)	(94.7)	96.7	98.2	(98.1)

Table 9.2: Final comparative table with AUROC scores between GANomaly (200 epochs), DRÆM (200 epochs), PaDiM (ResNet18 pre-trained), PatchCore (ResNet50V2 pre-trained), and GRD-Net (200 epochs). The values in parentheses represent pixel-level AUROC scores, while the dashes (-) indicate that the paper did not provide the respective data for those models. The results for GANomaly and DRÆM were obtained by adjusting the number of training epochs to match those used to train GRD-Net, which may cause slight deviations from the original reference paper.

In terms of surface anomaly detection, the proposed architecture not only boosts the final performance scores compared to the reference models but also smooths and steepens the learning curve, especially during the initial phase of training. Additionally, the gap between training and validation curves is noticeably smaller. The stability of the training process is enhanced by the GAN model, particularly due to the discriminator network.

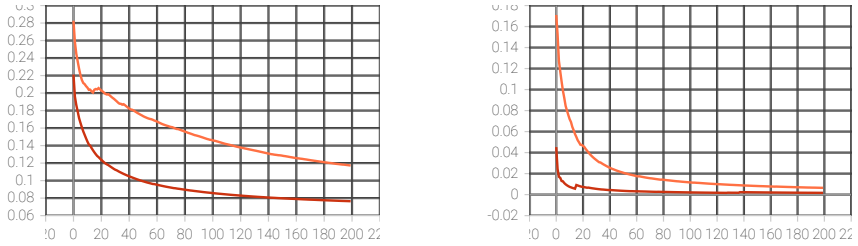
The sharper incline and reduced overfitting (as observed in the smaller difference between validation and training curves) can be attributed to the combination of the GAN model and the residual network. This improvement results from the adversarial nature of the GAN and the mitigation of gradient vanishing issues that often affect deep convolutional networks.

9.1.1.2 Anomaly Localization

Anomaly localization was also evaluated and compared with DRÆM after 200 training epochs. GANomaly was excluded from this comparison since the official paper does not provide a method for identifying defective regions. Table 9.2 presents the AUROC comparison between DRÆM and the proposed approach at four different stages of training: after 10, 50, 100, and 200 epochs.

The results are highly encouraging, surpassing those of the vanilla network. Improved image reconstruction quality directly contributes to the enhanced performance of the second, *discriminative* network. As seen in Figure 9.2, the validation curves for the proposed model (red) are significantly better compared to the vanilla model (orange), particularly in the early stages of training. This accelerates the process, reducing the number of epochs required to achieve acceptable performance for industrial applications.

On the other hand, *embedding similarity-based* models, such as PatchCore, tend to deliver better pixelwise AUROC scores. However, their architecture makes it difficult to incorporate an attention module based on ROIs.



(a) Validation contextual loss (b) Validation Focal Loss

Figure 9.2: Validation losses for generative (a) and discriminative (b) sub-networks. The red curve corresponds to the training of the proposed model, while the orange curve corresponds to the vanilla model. The learning curve is clearly improved in the case of the proposed model for both networks.

9.1.2 GRD-Net with ROI

Building on the previous experiment described in Paragraph 8.2.2.4.1, the ability of GRD-Net to effectively learn and focus on a ROI within an image for anomaly detection was further evaluated. The Zipper and Hazelnut datasets were used in this test. The Zipper dataset is particularly suitable for this evaluation, as it contains two distinct regions: the zipper itself and the fabric. As in the previous experiment, the zipper was chosen as the ROI to exclude defects in the fabric area, as shown in Figure 9.3.

As described earlier in Section 8.2.2.4, the *discriminative* network was trained using a focal loss applied to the intersection between the ROI and the mask generated from Perlin noise. This method enables the network to not only detect anomalies based on the differences between the original and reconstructed images but also to identify the specific areas where these differences are most likely to occur.

In many industrial scenarios, only certain regions of an image are relevant for defect detection. Including irrelevant regions can sometimes lead to false positives, as anomalies might appear in areas that do not pertain to the product under inspection.

To achieve these results, an ROI was defined for each training image, and the focal loss was tailored, as detailed in Paragraph 8.2.2.4, by intersecting the mask with the ROI during training. This allowed the *discriminative* network to focus its attention on the most relevant parts of the image, improving the model's ability to localize defects.

9.1.3 Ablation Study

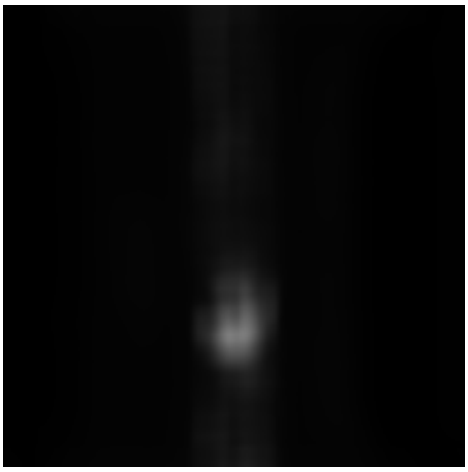
The GRD-Net architecture was analyzed by evaluating the network's *generative* model and the loss function used in the *discriminative* part.

9.1.3.1 Generative Model

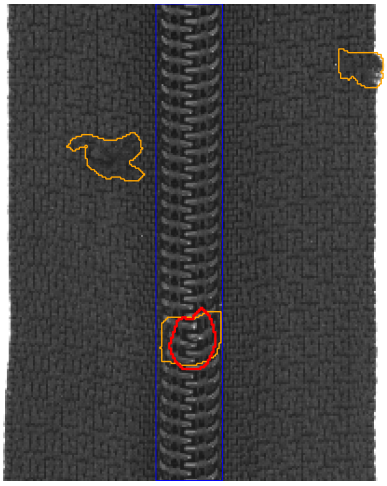
The *generative* sub-network, or the reconstructive part, was tested based on the state-of-the-art methods described in the DRÆM paper [170], particularly in Section 4.2, *Ablation Study - Architecture*. A GAN structure with a residual AE was introduced. Two configurations were tested: a *full-convolutional*

(a) Defect image (X)

(b) Ground truth (M)



(c) Generated *heatmap* by *discriminative* model after average pooling with 21×21 kernel



(d) Original image with regions: **blue** region is the ROI; the **orange** region is the ground truth (M); and the **red** region is the segmented region ($\hat{M}$)

Figure 9.3: An example from the zipper dataset showing three anomalies: one in the zipper, one in the middle of the fabric part, and another on the border of the fabric zone. Only the anomaly within the zipper region (inside the **blue** ROI) is detected, and it is almost perfectly aligned with the ground truth defect region. The **orange** region represents the ground truth, and the **red** region is the anomaly region segmented.

bottleneck with a latent size of $z = 32 \times 8 \times 8$ and a *dense bottleneck* with a latent size of $z = 2048$.

As previously shown, the best performance was achieved using the GAN architecture with a Convolutional bottleneck Residual Autoencoder (CRAE). On the other hand, the Dense bottleneck Residual Autoencoder (DRAE) performed well in tasks requiring anomaly removal but was less effective at learning random-like areas. A good example is the pill dataset, where pills exhibit a dotted, reddish texture that is better captured by the fully-convolutional bottleneck, as qualitatively shown in Figure 9.4.

(a) Original pill image (X)

(b) Pill image with Perlin noise $X_{(n)}$ ($\hat{X}$)

(c) Pill image rebuilt by GRD-Net with CRAE ($\hat{X}$)

(d) Pill image rebuilt by GRD-Net with DRAE ($\hat{X}$)

Figure 9.4: Pill dataset example: (a) Original image from the training set, (b) image with Perlin noise applied, (c) reconstruction from the CRAE, and (d) reconstruction from the DRAE. The CRAE shows superior performance, particularly in reproducing fine texture details.

9.1.3.2 Discriminative Model Loss

The original loss function for the *discriminative* model combined *Focal Loss* with *Cross-Entropy Overlap Distance Loss* [45, 46]. The rationale behind adding the second term was to encourage the network to focus on the ROI.

Initially, the loss was formulated as:

$$\mathcal{L}_{overlap}(\mathcal{A}_{discr}, \mathcal{ROI}_{input}) = w(1 - \frac{|\mathcal{A}_{discr} \cap \mathcal{ROI}_{input}|}{\min(|\mathcal{A}_{discr}, \mathcal{ROI}_{input}|)}), \tag{9.3}$$

where $w \in [0, 1]$, $\mathcal{L}_{overlap}$ is the contribution of the *Cross-Entropy Overlap Distance Loss* to the discriminative loss, $\mathcal{A}_{discr}$ represents the area mask generated by the discriminative network, and $\mathcal{ROI}_{input}$ is the reference ROI.

The complete loss function was then defined as:

$$\mathcal{L}_{FL} = \mathcal{FL}(\mathcal{A}_{discr}, \mathcal{M}_{input}) + \mathcal{L}_{overlap}(\mathcal{A}_{discr}, \mathcal{ROI}_{input}), \tag{9.4}$$

where $\mathcal{FL}$ is the focal loss and $\mathcal{M}_{input}$ is the input mask.

This formulation led the *discriminative* network to focus on the ROI, but it also resulted in highlighting the entire ROI in the generated heatmap. The term $\mathcal{A}_{discr}$ tended to approximate $\mathcal{ROI}_{input} \cdot w$, which was undesirable.

To address this issue, four variations of $\mathcal{L}_{FL}$ were tested:

1. The original formulation of Equation 9.4.
2. Vanilla *focal loss* using the intersection $\mathcal{I} = |\mathcal{A}_{discr} \cap \mathcal{ROI}_{input}|$ as input.
3. Vanilla focal loss with the overlap custom loss added.
4. A negated overlap function to avoid intersecting with $\mathbf{1} - \mathcal{ROI}_{input}$.

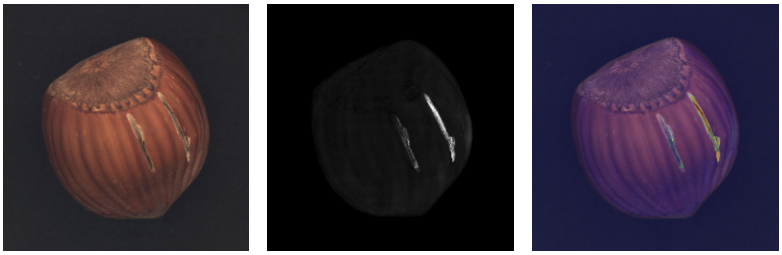
The best results, both visually (Figure 9.5) and numerically (Table 9.3), were obtained using Case 2, which tended to minimize $\min(\mathcal{A}_{discr} \cap \mathcal{ROI}_{input})$. Similar results were observed on the Zipper dataset, which was a good benchmark for real-case defects that appear both inside and outside the ROI, as shown in Figure 8.7.

	AUROC per image (pixel) after 100 epochs		
	AUROC	AUROC per pixel	Accuracy
Case 1	99.4	94.3	94.6
Case 2	**100.0**	**95.3**	**100.0**
Case 3	99.9	91.6	99.1
Case 4	**100.0**	93.5	**100.0**

Table 9.3: AUROC score after 100 epochs of training for each case, evaluated both per image and per pixel.

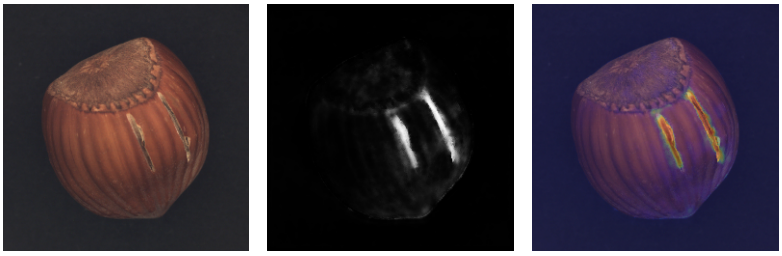
9.2 Real-Case Pharmaceutical Dataset

The proposed model was applied to a real-world industrial scenario for quality control on pharmaceutical BFS strips of vials. Tests were performed using an automatic machine by Bonfiglioli Engineering, which utilizes a rotary carousel



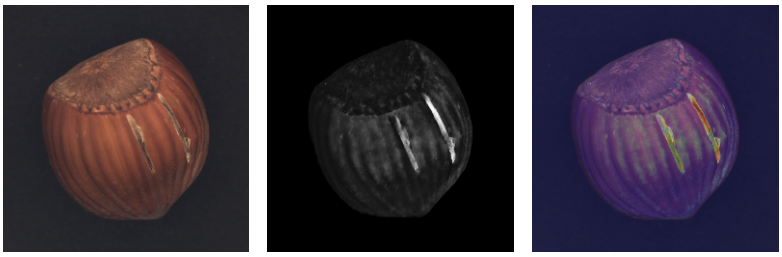
Case 1:

(a) Original image X (b) Generated heatmap $\hat{M}$ (c) Colorized and superimposed $\hat{M}$ on X



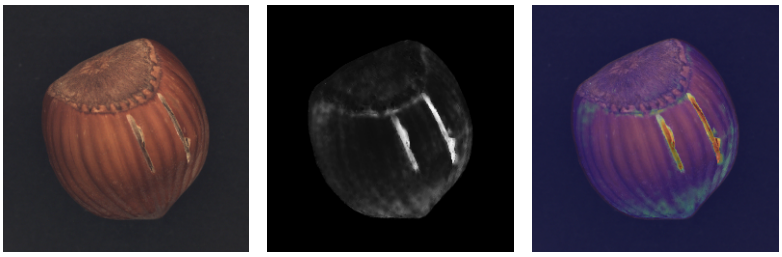
Case 2:

(d) Original image X (e) Generated heatmap $\hat{M}$ (f) Colorized and superimposed $\hat{M}$ on X



Case 3:

(g) Original image X (h) Generated heatmap $\hat{M}$ (i) Colorized and superimposed $\hat{M}$ on X



Case 4:

(j) Original image X (k) Generated heatmap $\hat{M}$ (l) Colorized and superimposed $\hat{M}$ on X

Figure 9.5: Visual comparison between the four loss functions for the *discriminative* network. The second case (method 2) yields the clearest segmentation of the anomalous areas.

equipped with sensors for image acquisition. The training set consisted of 230,355 images of vials, acquired in three different areas by an on-line camera during the production process.

Due to non-disclosure agreements, full product images cannot be shown. However, a limited area covering the most relevant part for the experiment is presented. The strips consist of five BFS plastic vials, adhered to one another along the sides and filled with liquid. One of the most challenging areas for analysis is the *meniscus* region, where the variability in shape, the presence of bubbles, or liquid drops over the meniscus make it difficult to process using traditional blob-analysis algorithms.

(a) Floating black particle on the meniscus, near the vial's *shoulder*

(b) Black spot near the meniscus

(c) Scratch at the turn of horizontal engraving

Figure 9.6: Three examples of real-world cases where traditional algorithmic analysis is difficult, if not nearly impossible.

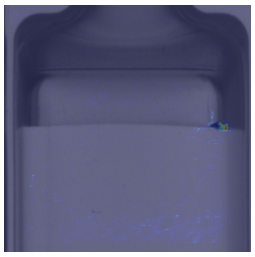
Figure 9.6 showcases three real-case examples where blob-analysis techniques struggle, primarily due to the variability in meniscus shape and the shadows created by the product's geometry and the sensor's position relative to the product.

	Best results after 30 epochs of training		
	Best AUROC per image	Best AUROC per pixel	Best accuracy
Vials in meniscus area	0.981	0.996	0.932

Table 9.4: Results of the real-case experiment after 30 epochs of training using GRD-Net with Case 2 loss explained in Section 9.1.3.2.

Case 1:

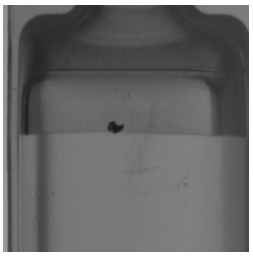
(a) Original image X with dark floating particle on meniscus



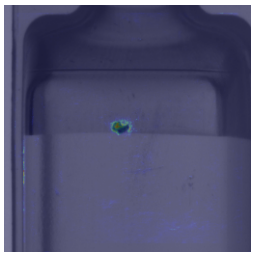
(b) Generated heatmap $\hat{M}$ normalized between 0 and 1

(c) Defect localization after convolution and threshold

Case 2:



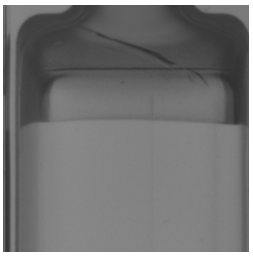
(d) Original image X with black spot on vial surface



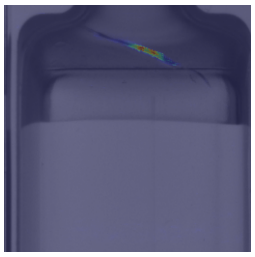
(e) Generated heatmap $\hat{M}$ normalized between 0 and 1

(f) Defect localization after convolution and threshold

Case 3:



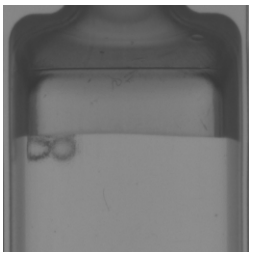
(g) Original image X



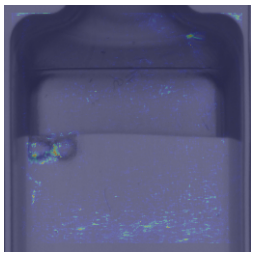
(h) Generated heatmap $\hat{M}$ normalized between 0 and 1

(i) Defect localization after convolution and threshold

Case 4:



(j) Original image X without defect but with bubbles near the meniscus



(k) Generated heatmap $\hat{M}$ normalized between 0 and 1

(l) Defect localization after convolution and threshold

Figure 9.7: Visual results from the real-case experiment. The first three images show defects, while the fourth shows a challenging regular product, difficult to detect using conventional methods.

With the proposed network, these anomalies were successfully localized, achieving results comparable to those obtained by human inspectors and classical algorithms. Figure 9.7 illustrates the results of the experiment, while Table 9.4 summarizes the performance statistics.

Chapter 10

Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line

Different experiments were carried out to optimize the network's performance and ensure low inference time, keeping it within the 500 ms slot between two acquisitions. To accurately describe the experiments, the background and hardware of the machine managing the products must be presented.

10.1 Product

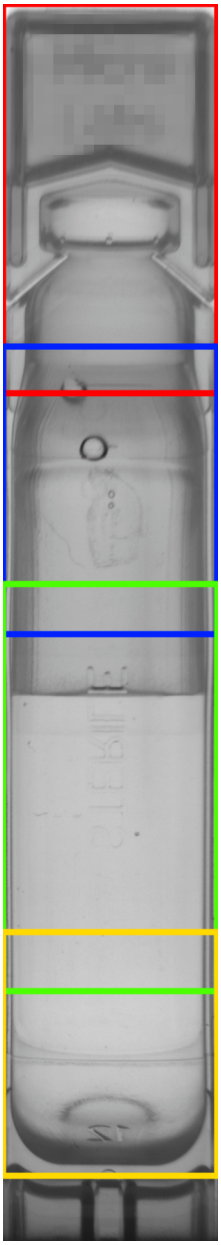
The product, displayed in Figure 10.1[1], is a 5-vial BFS strip filled with 10 ml of liquid excipient. Bubbles frequently form in the liquid, especially around the *meniscus*. Additionally, because the *neck* is thin, some liquid may get trapped inside it, and drops are commonly seen above the liquid level. As illustrated in Figure 10.2, each vial is divided into four logical regions: the *flag* (upper red region), the top *body* (blue region), the liquid *body* (green region), and the *bottom* (yellow region).

10.2 Training Dataset Acquisition

The acquisition setup was replicated on a laboratory scale using a telecentric lens, which minimized the distortion present in the side vials and bottom areas.

(a) The product

Figure 10.1: The product: a BFS strip composed of 5 vials[1].



(a) The vial regions

Figure 10.2: The logical regions into which each vial is divided[1].

Each product was captured 10 times following a semicircular movement focused on the upper part of the vial strip, resulting in 16 frames per acquisition. In addition, two extra images were generated using a *rank* filter that captured the lowest and highest gray values across the set of frames. According to the MVtec documentation[4]:

> The rank filter sorts gray values in ascending order within the mask and selects the gray value with rank *Rank*. Rank 1 corresponds to the smallest gray value, and rank A (in this case, $A = 16$) corresponds to the largest value.

As a result, anything that moved during handling appears in the darker images, while static regions remain in the lighter images. This technique increases variability in the features of positive images. The final dataset comprised 2,815,200 grayscale images ($256 \times 256 \times 1$), split into 90% training and 10% validation data.

10.3 Test Dataset Acquisition

A real-world set of defective strips was used to benchmark the algorithm's performance. Each acquisition consisted of three key frames (the first, eighth, and last) to reduce the system's computational load. The batch size (N) was calculated as:

$$N = f \cdot v \cdot r = 3 \cdot 5 \cdot 4 = 60, \tag{10.1}$$

where f represents the number of frames, v the number of vials, and r the number of regions.

10.4 Machine Handling

The machine used for inline quality control is placed at the end of the production line, after the filling machine and before the packaging station. Products are transferred from the conveyor belt into a carousel and handled identically

[4] `https://www.mvtec.com/doc/halcon/2305/en/rank_image.html`

to the laboratory mockup. Although some additional movement occurs before the carousel, past experiences indicate this extra motion does not significantly affect algorithm performance, as it is minimal compared to the movement generated at the inspection station. After handling, the product images are processed[5].

10.5 Experiments

Experiments were conducted over 10 epochs due to the high number of samples in the training set. The residual network follows the ResNet V2 architecture [64], and this section presents:

1. The hardware and software environment.
2. The architecture of the network.
3. The training process.
4. Accuracy on positive and negative test examples.
5. Results on time and throughput.
6. Experiments on the generative network.

10.5.1 Hardware

As already mentioned, the hardware used for training differs significantly from that used for inference. The training was performed on the following setup:

- **Hardware:**
  - Intel® Xeon® Silver 4216 CPU @ 2.10 GHz.
  - 64 GB DDR4 Synchronous @ 3200 MHz.
  - Nvidia® A100 GPU with 40 GB VRAM.
  - 2 TB M.2 NVMe SSD.
- **Software:**

[5]Further hardware details are withheld due to NDA and company policies.

  - Ubuntu 22.04.3 LTS with kernel 5.15.0-94-x86_64.
  - Nvidia® driver version 535.104.12.
  - CUDA® version 12.2.
  - TensorFlow 2.13.1 compiled from source.

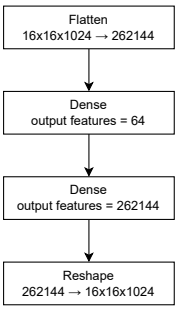
The inference machine, an industrial-grade cluster, was configured as follows:

- **Hardware:**
  - Intel® Xeon® E-2278GE CPU @ 3.30 GHz.
  - 32 GB DDR4 @ 3200 MHz.
  - Nvidia® A4500 GPU with 20 GB VRAM.
  - 32 GB CFast flash (read-only) for the OS.
  - 1 TB M.2 NVMe SSD for data.
- **Software:**
  - Ubuntu 22.04.3 LTS with kernel 5.15.0-94-x86_64.
  - Nvidia® driver version 535.104.12.
  - CUDA® version 12.2.
  - TensorFlow 2.13.1 compiled from source.

10.5.2 Network Architecture

10.5.2.0.1 Encoder As mentioned in subsection 8.2.2.5.1, the encoder is based on a residual network (ResNet V2). Each stage contains three residual blocks:

1. The first block concatenates three 3×3 convolutions with a 1×1 convolution, keeping the same input size.
2. The second block concatenates three 3×3 convolutions with the output of the first block, again keeping the same input size.



(a) Dense bottleneck

Figure 10.3: The bottleneck layer.

3. The third block concatenates three 3×3 convolutions with a downsampling operation that halves the height and width.

The network has four stages, reducing the input to a 16×16 embedding with 1024 filters.

10.5.2.0.2 Bottleneck The bottleneck layer is fully connected, with a size of 64 features, as shown in Figure 10.3.

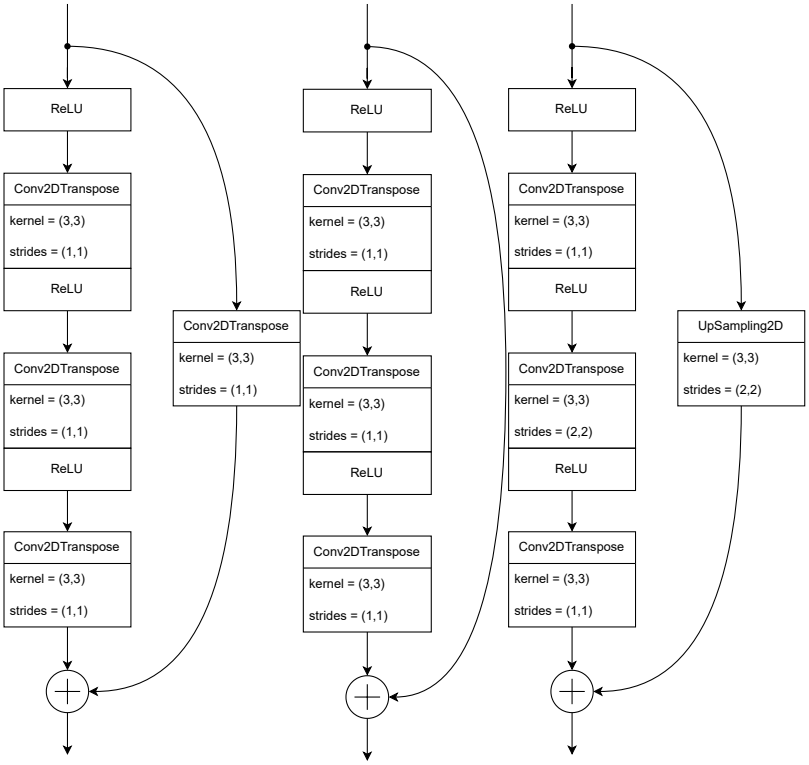
10.5.2.0.3 Decoder The decoder is the inverse of the encoder, using transposed convolutions to upsample the embeddings.

Each stage consists of three residual blocks, and the final output size is $256 \times 256 \times 1$, with a Sigmoid activation.

10.5.3 Training Phase

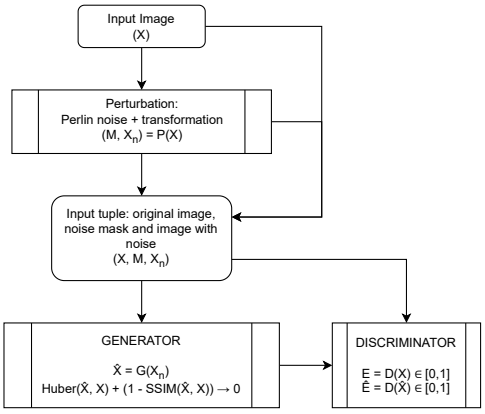
The model follows a GAN-like training process, alternating between the generator and discriminator. The learning rate starts at 1.5×10^{-4} and follows a cosine decay, restarting every 2,533,680 steps at one-third of the previous learning rate. The Adam optimizer is used for both networks, with a batch size of 32 and a 0.75 probability of applying perturbations to the input patches.

The reference flowchart is depicted in Figure 8.5 in Section 8.2.2.4. In this case, the flowchart, shown in Figure 10.5, is simplified due to the absence of the second network.



(a) ResDec Block A (b) ResDec Block B (c) ResDec Block C

Figure 10.4: Residual blocks in the decoder.



(a) Train flowchart

Figure 10.5: The training flowchart.

10.5.4 Results

Experiments were conducted using the client's *knapp test kit*, which contains 141 defective products and 120 regular ones. Since the dataset is balanced, accuracy was the main metric, along with true positive and true negative ratios and inference time. The results are broken down as follows:

1. Overall accuracy and inference time per patch.
2. Accuracy after image aggregation and inference time per product.
3. Accuracy per product and run, based on the client's acceptance policy.

Some images are presented in Appendix E.

10.5.4.0.1 Acceptance Policy For validation, the machine must match or outperform human operators on the *knapp test kit*. Each product is acquired 10 times, and it is correctly classified if 7 out of 10 runs result in the correct label. Similarly, defective products are correctly classified if they receive 7 negative classifications.

10.5.4.0.2 Overall Results Table 10.1 presents the overall results, with a threshold calculated for each region. The mean inference time is also calculated as:

$$\mu_{t_f} = \frac{\mu_{t_b}}{f \cdot v \cdot r} = \frac{\mu_{t_b}}{60}, \tag{10.2}$$

where μ_{t_f} is the mean frame time, μ_{t_b} is the mean batch time, v is the number of vials, and r is the number of regions per vial.

10.5.4.0.3 Per Product Aggregation The classification of a vial strip is based on the most severe label; if any region is rejected, the entire product is rejected. This reduces the false positive ratio but increases the false rejection ratio, requiring new threshold tuning. The final anomaly score for each vial is given by:

$$\theta_v = \max\left(1.0 - \text{SSIM}(X_i, \hat{X}_i)\right), \quad 0 \leq i < f \cdot v \cdot r, \tag{10.3}$$

Overall results					
	R0	**R1**	**R2**	**R3**	**Vial**
Threshold	0.015589	0.038017	0.046568	0.029593	-
Accuracy	0.9919	0.9926	0.9957	0.9991	0.9870
True positive ratio	0.9966	0.9985	0.9986	0.9994	0.9942
True negative ratio	0.9093	0.9044	0.9779	0.9973	0.9581
Mean inference time (ms)	0.1689	0.1689	0.1689	0.1689	-

Table 10.1: Summary of overall results using Equation 8.14 as the anomaly score across different regions: R0 (top region), R1 (shoulder region), R2 (liquid region), R3 (bottom region), and across single vials in the strip.

Per strip results	
	Score
Threshold R0	0.016156
Threshold R1	0.038584
Threshold R2	0.046635
Threshold R3	0.029660
Accuracy	0.9593
True positive ratio	0.9694
True negative ratio	0.9467
Mean inference time	0.4873

Table 10.2: Per strip results using the equation 10.3 as anomaly score.

where i is the patch index.

10.5.4.0.4 Per Run Aggregation For validation, the machine must outperform the client's manual inspectors on the *knapp test kit*. Each product is acquired 10 times, and a product is considered correctly classified if it gets 7 out of 10 consistent classifications. Table 10.3 shows the results.

10.5.5 Conclusion on Results

The results significantly exceeded expectations, not only making feasible what would be unmanageable with traditional methods but also surpassing the client's requirements. The model performed within the target time constraints set by the automation, demonstrating high classification accuracy and efficient defect localization. These outcomes validate the integration of the deep genera-

Per run results	
	Score
Threshold R0	0.013222
Threshold R1	0.034650
Threshold R2	0.046201
Threshold R3	0.029226
Accuracy	0.9641
True positive ratio	0.9676
True negative ratio	0.9599

Table 10.3: Per run results using the equation 10.3 as anomaly score.

tive anomaly detection algorithm into the high-speed industrial line, providing a reliable solution for quality control.

Part V

Conclusions and Outlooks

Chapter 11

Conclusions and Future Works

11.1 Conclusions

This thesis presents a comprehensive investigation into Deep Learning (DL) models for Anomaly Detection (AD) in industrial settings, particularly focusing on Visual Inspection (VI) tasks in high-speed production lines. The research presents four primary contributions: the development of a classification-based AD system, the design and implementation of Generative Reconstructive Discriminative Network (GRD-Net) [43], Residual Generative Adversarial Network for Anomaly Detection (ResGANAD) [44] and its industrial integration, and the exploration of Mahalanobis distance-based techniques culminating in the novel Mahalanobis-Based Clusters PatchCore (MhBC-PatchCore). This model, based on embedding similarity architectures like PatchCore, address specific challenges within the field, advancing the automation and accuracy of QC processes.

In particular, QC processes, especially within the pharmaceutical industry, hold not only a business impact but also a significant Good Manufacturing Practice (GMP) impact, as the quality and integrity of the products directly affect patient health. Ensuring stringent GMP compliance is crucial, as any undetected anomalies in pharmaceutical products could pose serious risks to end-users. Thus, the ability of these DL models to enhance anomaly detection and improve product reliability is of paramount importance, reinforcing the critical role of automated quality control in safeguarding public health.

The first major contribution involved the implementation of a classification-

based AD system, where the model was trained to distinguish between defective and non-defective products using a supervised approach. While effective in detecting anomalies in controlled environments, this model encountered limitations when applied to real-world data with high variability. Subtle defects or anomalies, which were rare in the training data, proved difficult to detect, underscoring the need for more flexible and generalizable models capable of handling unseen defects. Although this system provided a solid foundation for AD, it highlighted the limitations of traditional classification-based approaches and demonstrated the necessity of more sophisticated techniques for addressing rare or subtle defects.

The second major contribution, and arguably the most significant of this thesis, is the development of GRD-Net. This novel architecture combines both generative and discriminative components, leveraging reconstruction-based and classification-based strategies to detect anomalies. The generative sub-network, inspired by state-of-the-art residual architectures and trained using a Generative Adversarial Network (GAN) framework, focused on reconstructing normal samples. The discriminative part, on the other hand, identified anomalies by analyzing the discrepancies between the original and reconstructed data, particularly within a Region of Interest (ROI) provided during the training phase. This approach proved highly effective in detecting anomalies, especially in complex and highly variable scenarios. For instance, in the context of pharmaceutical Blow Fill Seal (BFS) strip inspections, the model excelled in detecting anomalies in the meniscus region, an area where traditional blob analysis algorithms struggled due to the inherent variability in the liquid surface and the presence of bubbles or liquid drops. GRD-Net's ability to handle such challenging defects demonstrates the robustness of its architecture, which was designed to capture both local and global features, making it adaptable to a wide range of industrial applications.

The third significant contribution of this work is the successful integration of the ResGANAD model into a real-time industrial environment. This architecture, optimized and derived from GRD-Net, was adapted to meet the stringent real-time processing requirements of high-speed production lines, where each sample must be analyzed within a window of less than 500 ms. This research tackled the challenges of optimizing both the software and hardware

environments to enable the deployment of DL models in real-time without compromising accuracy. By leveraging high-performance GPUs, efficient data handling techniques, and optimized inference times, the system was able to meet these strict time constraints while maintaining high detection accuracy. The integration of ResGANAD into a production line for real-time VI marks a critical achievement in demonstrating the practical applicability and scalability of DL models in industrial environments. The model's ability to detect defects with high accuracy in real-time provides a reliable solution for automated Quality Control (QC) in industries where speed and precision are critical.

The fourth and final contribution involves the exploration and implementation of Mahalanobis-based methods, specifically MhBC-PatchCore, which introduced a novel dimension to AD by analyzing feature space distances. While GRD-Net excelled in generative reconstruction, MhBC-PatchCore focused on detecting anomalies by measuring how far a patch of an image deviated from the expected distribution of normal patches using the Mahalanobis distance. This approach provided a complementary solution to generative models, excelling in cases where the underlying distribution is a multivariate Gaussian. By analyzing the deviation at the feature level, rather than relying on reconstruction error, MhBC-PatchCore achieved higher accuracy in identifying anomalies across a diverse set of products compared to reference architectures such as PatchCore and PaDiM. Furthermore, it represents a feasible solution for handling large-scale datasets in industrial settings. However, the architecture's limitations lie in its dependency on pre-trained large models, such as ResNets, which may exhibit domain mismatch when applied to specific industrial datasets. This mismatch can weaken the model's ability to effectively capture domain-specific features, requiring further consideration of domain adaptation techniques for future work.

Through these four key works, this thesis has provided significant advancements in the field of industrial AD, providing both theoretical and practical contributions. The research demonstrates how DL models, ranging from classification-based systems to reconstruction-based and embedding similarity-based approaches, can be effectively integrated into industrial environments to improve automated QC processes. The ability to detect anomalies with high

accuracy, even in complex and variable conditions, marks a substantial step forward in the domain of automated Visual Inspection (VI).

In conclusion, this thesis contributes to the growing body of research on DL for industrial AD, highlighting the effectiveness and adaptability of these models in real-world applications. By addressing the limitations of traditional AD algorithms and presenting scalable, real-time solutions, the work presented in this thesis lays a solid foundation for the future development of intelligent, automated QC systems. The successful integration of these models into high-speed production lines represents a crucial step towards the broader adoption of AI-driven technologies in industrial environments, offering the potential for increased efficiency, accuracy, and scalability in automated inspection tasks.

11.2 Future Work

The field of AD in industrial applications remains a critical area of research due to its essential role in automating QC processes, which are vital for ensuring efficiency and safety. One of the longstanding challenges in this domain is the very definition of what constitutes an anomaly, as this determination is often highly contextual and subjective. With the development of increasingly sophisticated DL models, the conversation has shifted towards a more practical and effective approach to Out-Of-Distribution Anomaly Detection (OOD-AD).

In recent years, a significant number of architectures have been developed to tackle this task, creating distinct families of models that share common principles. The most established approaches in the literature include reconstruction-based models, embedding similarity-based models, and the notable subcategory of normalizing flow-based methods. Another solidified approach is knowledge distillation-based models. All of these techniques rely on the assumption that the number of normal (or “good”) samples far exceeds the number of defect or anomaly cases. Consequently, the learning process focuses primarily on the "good" class, while the few anomaly instances are used primarily to optimize a threshold that governs the chosen performance metric.

A promising and rapidly evolving family of models for OOD-AD is Synthetic Data Generation. This process involves creating artificial data that mimics the statistical patterns and properties of real-world data through vari-

ous algorithms and models. While synthetic data is often based on real data, it typically contains no actual values from the original dataset. With the advent of diffusion models [5, 151, 81, 72, 73, 118] and neural networks that implement multi-head attention and self-attention layers [159], researchers have significantly surpassed the performance of traditional generative models such as AEs, VAEs, GANs, and WGANs.

In addition, works like GLASS [98] have introduced novel techniques for generating synthetic anomalous data while remaining within the "perceptual vicinity" of normal data. These approaches suggest a future direction in which architectures could be developed that generate synthetic data by combining various generation techniques with a binary pixel-wise classifier. This classifier would be capable of generating a final heatmap that highlights anomalous regions at the pixel level. Such a classifier could also potentially overcome the limitations of conventional CNNs, which maintain a strict correlation between the position of a pixel region in the input image and the corresponding embedding vector in a deep layer of the network. Despite the well-known computational efficiency of CNNs, this spatial rigidity limits the ability to model complex relationships between features within the image.

By contrast, models based on the concept of self-attention, such as ViTs, break away from this spatial constraint, allowing for more flexible and comprehensive feature interactions. Mixed approaches, such as those found in the VQVAE [41] architecture, leverage transformer-based technologies to handle large-scale images with sustainable hardware requirements. These advancements are pushing research towards a deeper commitment to architectures that combine the power of transformers and CNNs with the flexibility of AD models.

The future direction of AD may, therefore, involve the creation of architectures capable of generating synthetic data that is both perceptually and statistically similar to real-world anomalies. By mixing techniques like diffusion models, attention mechanisms, and pixel-wise binary classifiers, these systems could provide highly accurate and interpretable AD solutions. In particular, the use of self-attention mechanisms in conjunction with synthetic data generation opens the door to new possibilities in terms of both accuracy and scalability, especially when integrated into high-speed industrial environments.

As research continues to evolve, the development of more powerful and efficient models will likely drive the next wave of advancements in industrial anomaly detection, offering even more robust solutions to QC challenges.

Appendix

Appendix A

Detailed Explanation of Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) [144] is a widely adopted technique designed to enhance the interpretability of Convolutional Neural Networks (CNNs) by providing visual explanations for the model's classification decisions. This method is particularly useful in scenarios where understanding the rationale behind a model's predictions is as crucial as the predictions themselves, such as in medical diagnostics, autonomous driving, and the pharmaceutical industry.

A.1 Theoretical Foundation

At its core, Grad-CAM leverages the gradient information flowing into the last convolutional layer of a CNN to understand the importance of each neuron for a specific decision. Unlike fully connected layers, which tend to lose spatial information, convolutional layers retain the spatial hierarchy of the input, making them ideal for visualizing which parts of the input contribute most to the decision.

Let y^c denote the score for class c (before applying the softmax) and A^k represent the k-th feature map (activation map) in a convolutional layer of the network. The core idea of Grad-CAM is to compute the gradient of y^c with respect to the feature map A^k, which indicates how much the activation in each location of A^k contributes to the class score y^c. Mathematically, this

gradient is expressed as:

$$\frac{\partial y^c}{\partial A^k} \tag{A.1}$$

This gradient information reflects the importance of the feature map with respect to the class under consideration.

A.2 Importance Weights and Heatmap Generation

To aggregate this information across the entire feature map, Grad-CAM performs a global average pooling over the spatial dimensions (width and height) of the gradient, resulting in a set of importance weights α_k^c for each feature map k:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{A.2}$$

Here, Z represents the total number of pixels in the feature map A^k, and the indices i and j run over the spatial dimensions of the feature map. The importance weight α_k^c signifies the relative importance of the feature map A^k for the class c.

The next step involves combining these weights with the corresponding feature maps to produce a weighted activation map. This is achieved by performing a weighted sum of the feature maps:

$$L_{\text{Grad-CAM}}^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right) \tag{A.3}$$

The use of the ReLU activation function ensures that only the positive contributions to the class score are considered, effectively filtering out any negative influences, which might detract from the class score. The resulting

heatmap $L^c_{\text{Grad-CAM}}$ highlights the regions of the input image that are most relevant to the network's prediction for class c.

A.3 Interpretability and Applications

Grad-CAM offers a powerful tool for making CNN-based models more interpretable. By overlaying the heatmap on the original input image, practitioners can visually assess which parts of the image influenced the model's decision, thereby gaining insights into the model's inner workings. This capability is particularly valuable in domains where decisions must be transparent and justifiable.

For example, in the pharmaceutical industry, where ML models might be used to detect anomalies in medical images or to classify compounds based on their potential efficacy, understanding the "why" behind a model's decision is critical. Misinterpretations can lead to incorrect diagnoses or the misclassification of potentially life-saving drugs. By employing Grad-CAM, researchers and clinicians can verify that the model is focusing on the correct regions of an image—such as the active site of a compound or a specific anomaly in a medical scan—thus building trust in the model's predictions [144].

A.4 Limitations and Extensions

While Grad-CAM is a significant step towards interpretable AI, it is not without its limitations. The resolution of the heatmaps generated by Grad-CAM is constrained by the resolution of the feature maps, which are often downsampled relative to the input image. This can result in coarse visualizations that might not capture fine-grained details. Additionally, Grad-CAM assumes that positive gradients correspond to positive contributions to the class score, which might not always hold true in more complex networks.

Extensions of Grad-CAM, such as Guided Grad-CAM and Smooth Grad-CAM++, aim to address some of these limitations by combining gradient information with guided backpropagation or by smoothing the gradients, thereby producing more detailed and reliable visual explanations.

Grad-CAM remains an essential tool for the interpretable application of

deep learning models in critical fields, facilitating the alignment of model predictions with domain knowledge and human intuition.

Appendix B

Detailed Explanation of Mahalanobis Distance

B.1 Multivariate Gaussian Process

A multivariate Gaussian process is an extension of the Gaussian distribution to multiple variables. In this context, a random vector $\mathbf{x}$ is said to follow a multivariate Gaussian distribution if it is defined by:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{B.1}$$

where:

- $\boldsymbol{\mu}$ is the mean vector of dimension d,
- $\boldsymbol{\Sigma}$ is the $d \times d$ covariance matrix.

This process describes the joint distribution of multiple variables, where $\boldsymbol{\Sigma}$ encodes the relationships (i.e., covariances) between the variables. A multivariate Gaussian process provides a probabilistic framework for modeling data distributions and detecting deviations from the norm.

B.2 Covariance Matrix

The covariance matrix $\boldsymbol{\Sigma}$ represents the degree to which two random variables vary together. It is mathematically defined as:

$$\boldsymbol{\Sigma} = \mathbb{E}\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\right] \tag{B.2}$$

In practice, the covariance matrix can be estimated from a set of N observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ of the random vector $\mathbf{x}$. The sample covariance matrix is given by:

$$\Sigma = \frac{1}{N-1}\sum_{k=1}^{N}(\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_k - \boldsymbol{\mu})^T \tag{B.3}$$

where $\boldsymbol{\mu}$ is the mean of the observations, and N is the number of samples. The covariance matrix plays a critical role in measuring relationships between variables and is essential for computing the Mahalanobis distance.

B.3 Mahalanobis Distance

The Mahalanobis distance is a measure of the distance between a point and a distribution, taking into account the correlations between variables. It is defined as:

$$d_M(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{B.4}$$

where:

- $\mathbf{x}$ is the point being evaluated,
- $\boldsymbol{\mu}$ is the mean vector of the distribution,
- $\boldsymbol{\Sigma}$ is the covariance matrix.

The Mahalanobis distance is particularly useful in scenarios where the data exhibits correlations between variables, as it scales the distance based on the variability in different directions. The inverse of the covariance matrix $\mathbf{\Sigma}^{-1}$ accounts for these correlations.

B.4 Application to Embedding-Based Systems

In certain use cases, such as in computer vision tasks, embeddings are extracted from a pretrained neural network to represent patches of an image. In this scenario, each patch (i, j) of an image is represented by an embedding vector x_{ij}, and for each patch position (i, j), a mean vector μ_{ij} and a covariance matrix Σ_{ij} are learned from the training data.

During the training phase, embeddings are extracted for each patch from multiple layers of the neural network. These embeddings are then used to calculate the mean vector μ_{ij} and the covariance matrix Σ_{ij} for each patch position (i, j), where:

$$\Sigma_{ij} = \frac{1}{N-1} \sum_{k=1}^{N} (x_{ij}^k - \mu_{ij})(x_{ij}^k - \mu_{ij})^T + \epsilon I \tag{B.5}$$

Here:

- x_{ij}^k is the embedding vector for patch (i, j) in the k-th training image,
- μ_{ij} is the mean vector for the embeddings of patch (i, j),
- ϵI is a regularization term added to the covariance matrix to ensure invertibility.

At inference time, for a test image, the Mahalanobis distance is computed for each patch by comparing the embedding vector x_{ij} of the patch to the mean vector μ_{ij} and covariance matrix Σ_{ij} learned during training. The Mahalanobis distance for patch (i, j) is given by:

$$M(x_{ij}) = \sqrt{(x_{ij} - \mu_{ij})^T \Sigma_{ij}^{-1} (x_{ij} - \mu_{ij})} \tag{B.6}$$

This distance is calculated for each patch, and a matrix M is constructed that contains the Mahalanobis distances for all patches in the image:

$$M = (M(x_{ij}))_{1 \leq i \leq W, 1 \leq j \leq H} \tag{B.7}$$

where $W \times H$ represents the resolution of the grid of patches. This matrix of Mahalanobis distances can be used to detect anomalous patches in the image, where higher values of $M(x_{ij})$ indicate regions that deviate significantly from the learned normal distribution.

B.5 Generalized Mahalanobis Distance Between Two Points

The Mahalanobis distance can be generalized to measure the distance between two points, $\mathbf{x}_1$ and $\mathbf{x}_2$, rather than between a point and the mean of a distribution. The generalized Mahalanobis distance is given by:

$$d_M(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{\Sigma}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)} \tag{B.8}$$

This formulation considers the covariance matrix $\mathbf{\Sigma}$ to account for the relationships between variables when computing the distance between two points. It provides a robust way of comparing points in a multivariate space, adjusting for correlations in the data.

B.6 Covariance Matrix Regularization

In practice, the covariance matrix $\mathbf{\Sigma}$ must be invertible to compute the Mahalanobis distance. To ensure that the matrix is invertible, a regularization term is often added:

$$\Sigma_{\text{regularized}} = \Sigma + \epsilon I \tag{B.9}$$

where ϵ is a small positive constant, and I is the identity matrix. This regularization ensures that the covariance matrix remains positive definite, allowing for a stable and reliable computation of the Mahalanobis distance.

Appendix C

Online Covariance Calculation

C.1 Introduction

In situations where datasets are too large to fit into memory, it becomes necessary to compute statistics like covariance in an online or incremental manner. The class `OnlineCovariance` provides a solution for calculating the covariance matrix incrementally as new batches of data are introduced, without needing to store the entire dataset in memory. This appendix explains how the online covariance calculation works and provides a step-by-step explanation of the code implementation.

C.2 Code Explanation

The class `OnlineCovariance` is designed to compute the covariance matrix for datasets too large to fit into memory. The covariance matrix is updated incrementally as batches of data are processed. Below is an explanation of the class methods.

C.2.1 Class Initialization

The constructor method `__init__` initializes the necessary variables:

- `dim`: The dimensionality of the input data.
- `self.n`: The number of data points processed so far, initialized to zero.

- `self.sum_x`: A vector initialized to zero that will accumulate the sum of all data points processed. It is a zero vector of size `dim`.
- `self.sum_x2`: A matrix initialized to zero that will accumulate the sum of the outer products of the data points. It is a zero matrix of shape [`dim`, `dim`].

C.2.2 Update Method

The `update` method processes a new batch of data and updates the internal statistics accordingly:

- `data`: A batch of data points with shape [`batch_size`, `dim`].
- The method updates `self.n` by adding the batch size (i.e., the number of new data points).
- It updates `self.sum_x` by adding the sum of the current batch of data points across the batch dimension.
- It updates `self.sum_x2` by computing the sum of the outer products of the batch data points and adding it to the current `self.sum_x2`.

C.2.3 Covariance Calculation Method

The `get_covariance` method computes and returns the current covariance matrix using the accumulated statistics:

- The mean vector μ is calculated as:

$$\mu = \frac{\texttt{sum\_x}}{n} \tag{C.1}$$

- The covariance matrix is calculated as:

$$\text{cov} = \frac{\texttt{sum\_x2}}{n} - \mu\mu^T \tag{C.2}$$

- The method returns this covariance matrix, which reflects all data processed up to this point.

C.3 Complete Code in Python

Here is the complete Python code for the `OnlineCovariance` class that performs online covariance calculation.

```
class OnlineCovariance:
        """
        This class calculates the covariance of a dataset in an online manner.
        It is useful when the entire dataset cannot fit into memory.
        """

        def __init__(self, dim):
                """
                Initialize the OnlineCovariance object.

                Args:
                dim (int): The dimensionality of the data.
                """
                # The number of data points seen so far.
                self.dim = dim
                self.n = 0

                # The sum of the data points.
                self.sum_x = tf.zeros([dim], dtype=tf.float64)

                # The sum of the outer product of the data points.
                self.sum_x2 = tf.zeros([dim, dim], dtype=tf.float64)

        def update(self, data):
                """
                Update the statistics with a new batch of data.

                Args:
                data (tf.Tensor): A batch of data points. The shape is [batch_size, dim].
                """
                # Update the count.
                self.n += tf.shape(data)[0]

                # Update the sum.
                self.sum_x += tf.reduce_sum(data, axis=0)

                # Update the sum of outer products.
                self.sum_x2 += tf.linalg.matmul(tf.transpose(data), data)

        def get_covariance(self):
                """
                Calculate and return the covariance matrix of the data seen so far.

                Returns:
                tf.Tensor: The covariance matrix. The shape is [dim, dim].
                """
                # Calculate the mean.
                mean = self.sum_x / tf.cast(self.n, dtype=tf.float64)

                # Calculate the covariance.
                cov = self.sum_x2 / tf.cast(self.n, dtype=tf.float64) - tf.tensordot(mean, mean, axes=0)
                return cov
```

C.4 Algorithm for Online Covariance Calculation

The online covariance calculation can be described using the following algorithm:

Algorithm 1 Online Covariance Calculation

Input: data points $\mathbf{X}$, dimension d
Initialize:
$\mathbf{sum\_x} \leftarrow \mathbf{0}$ ▷ Sum of data points
$\mathbf{sum\_x2} \leftarrow \mathbf{0}$ ▷ Sum of outer products
$n \leftarrow 0$ ▷ Count of data points
procedure UPDATE($\mathbf{X}_{\text{batch}}$)
    $n \leftarrow n + \text{batch\_size}$
    $\mathbf{sum\_x} \leftarrow \mathbf{sum\_x} + \sum \mathbf{X}_{\text{batch}}$
    $\mathbf{sum\_x2} \leftarrow \mathbf{sum\_x2} + \mathbf{X}_{\text{batch}}^T \mathbf{X}_{\text{batch}}$
end procedure
procedure GETCOVARIANCE
    $\mu \leftarrow \mathbf{sum\_x}/n$ ▷ Calculate mean vector
    Return: $\mathbf{cov} \leftarrow \mathbf{sum\_x2}/n - \mu\mu^T$
end procedure

This algorithm processes batches of data, updating the sum of the data points and the sum of the outer products to calculate the covariance matrix without needing to store the entire dataset in memory.

Appendix D

Cholesky Decomposition and Mahalanobis Distance

D.1 Cholesky Decomposition

The Cholesky decomposition is a matrix factorization technique used for symmetric, positive-definite matrices. Given a symmetric, positive-definite matrix $\mathbf{\Sigma}$, its Cholesky decomposition is:

$$\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^T \tag{D.1}$$

where:

- $\mathbf{L}$ is a lower triangular matrix,
- $\mathbf{L}^T$ is the transpose (upper triangular matrix) of $\mathbf{L}$.

The Cholesky decomposition is particularly useful for computational efficiency when solving systems of linear equations or when computing the inverse of a matrix. In this context, it also simplifies the computation of the Mahalanobis distance by transforming the space in which the distance is computed.

D.2 Mahalanobis Distance via Cholesky Decomposition

The Mahalanobis distance, as defined in Appendix B, is:

$$d_M(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{D.2}$$

Using the Cholesky decomposition $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$, the Mahalanobis distance can be rewritten. First, consider the transformation:

$$\mathbf{z} = \mathbf{L}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \tag{D.3}$$

This transformation maps the vector $\mathbf{x}$ into a space where the covariance matrix becomes the identity matrix. Substituting this into the Mahalanobis distance formula yields:

$$d_M(\mathbf{x}, \boldsymbol{\mu}) = \|\mathbf{z}\|_2 = \sqrt{\mathbf{z}^T \mathbf{z}} = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{D.4}$$

Thus, the Mahalanobis distance in the transformed space can be seen as the Euclidean distance in the space where the covariance matrix has been "whitened" by the Cholesky decomposition.

D.2.1 Explicit Form of Mahalanobis Distance

The form of $\sqrt{\mathbf{z}^T \mathbf{z}}$ can be further expanded using the Cholesky decomposition. Recall that:

$$\mathbf{z} = \mathbf{L}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \tag{D.5}$$

Substituting this back into the Mahalanobis distance, the following expres-

sion is obtained:

$$d_M(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{\left(\mathbf{L}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)^T \left(\mathbf{L}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)} \tag{D.6}$$

Since the Cholesky decomposition ensures that $\boldsymbol{\Sigma}^{-1} = \mathbf{L}^{-T}\mathbf{L}^{-1}$, this form simplifies to:

$$d_M(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{L}^{-T}\mathbf{L}^{-1}(\mathbf{x} - \boldsymbol{\mu})} \tag{D.7}$$

This highlights how the Cholesky decomposition simplifies the computation of the Mahalanobis distance by decomposing the covariance matrix into a product of lower triangular matrices, making it computationally efficient to work with $\mathbf{L}$ and $\mathbf{L}^T$.

D.3 Regularized Cholesky Decomposition

In practical applications, especially when working with high-dimensional data or when the covariance matrix is nearly singular, the covariance matrix $\boldsymbol{\Sigma}$ may not be invertible. To address this issue, a regularization term is added, similar to the regularization described in Appendix B. The regularized covariance matrix is:

$$\boldsymbol{\Sigma}_{\text{regularized}} = \boldsymbol{\Sigma} + \epsilon I \tag{D.8}$$

where ϵ is a small positive constant and I is the identity matrix. The Cholesky decomposition is then applied to the regularized covariance matrix:

$$\boldsymbol{\Sigma}_{\text{regularized}} = \mathbf{L}_{\text{regularized}}\mathbf{L}^T_{\text{regularized}} \tag{D.9}$$

This ensures that the covariance matrix remains positive definite, making the Cholesky decomposition possible and allowing for stable computation of

the Mahalanobis distance.

D.4 Mahalanobis Distance for Transformed Embeddings

Applying the Cholesky decomposition in the context of transformed embeddings, such as in machine learning applications, allows for the computation of the Mahalanobis distance for the transformed embeddings $\mathbf{z} = \mathbf{L}\mathbf{x}$. After applying the Cholesky decomposition, the Mahalanobis distance becomes:

$$d_M(\mathbf{z}, \boldsymbol{\mu}) = \|\mathbf{z} - \mathbf{L}\boldsymbol{\mu}\|_2 \tag{D.10}$$

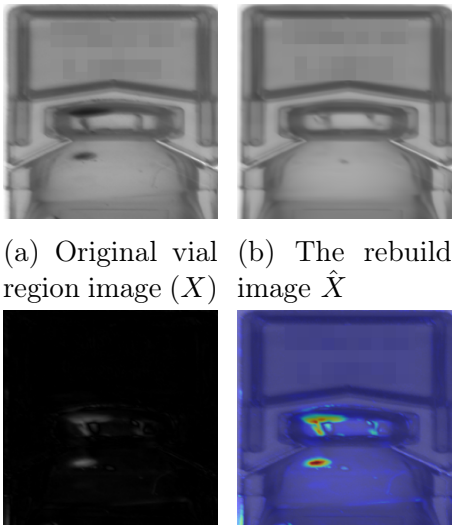
This representation simplifies the computation of the Mahalanobis distance and ensures numerical stability through the regularized covariance matrix.

Appendix E

Large-Scale Pharmaceutical Dataset Result Images

E.1 Result Images

Below are shown examples taken from real cases analyzed during algorithm validation phase.



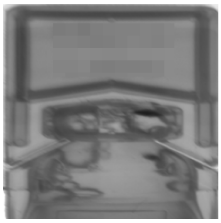
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

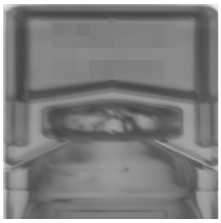
(c) The difference image

(d) The heatmap H superimposed

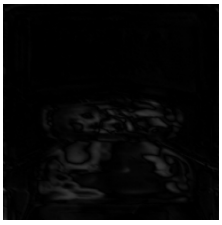
Figure E.1: Stuck particle in the upper part of the product



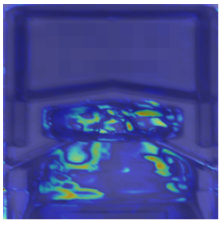
(a) Original vial region image (X)



(b) The rebuild image $\hat{X}$

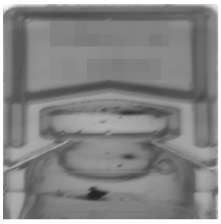


(c) The difference image



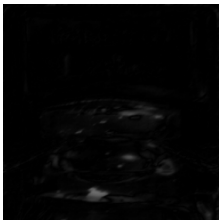
(d) The heatmap H superimposed

Figure E.2: Stuck particle and anomalous liquid behavior that results in foam formation because of contamination

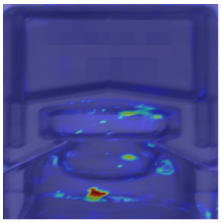


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

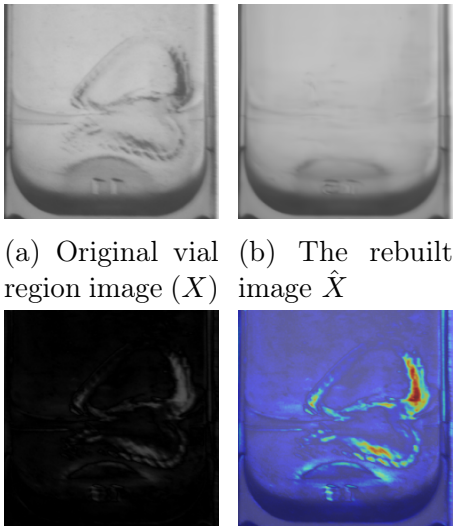


(c) The difference image



(d) The heatmap H superimposed

Figure E.3: Stuck particle and anomalous liquid behavior that results in foam formation because of contamination



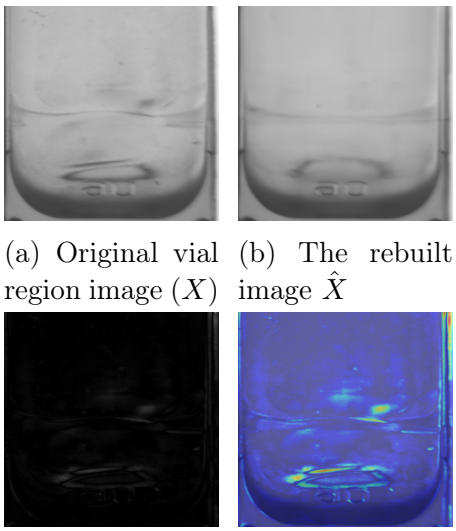
(a) Original vial region image (X)

(b) The rebuilt image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.4: Lower body deformation



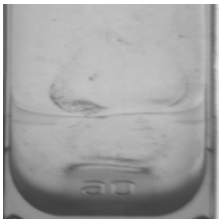
(a) Original vial region image (X)

(b) The rebuilt image $\hat{X}$

(c) The difference image

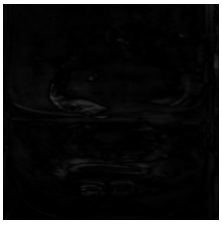
(d) The heatmap H superimposed

Figure E.5: Light lower body deformation

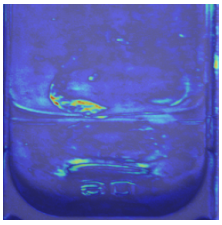


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

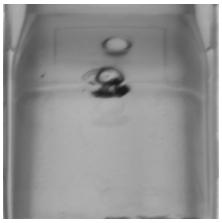


(c) The difference image



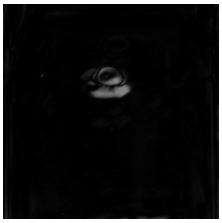
(d) The heatmap H superimposed

Figure E.6: Lower body deformation

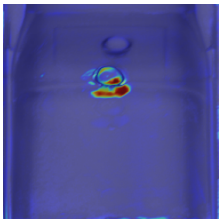


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

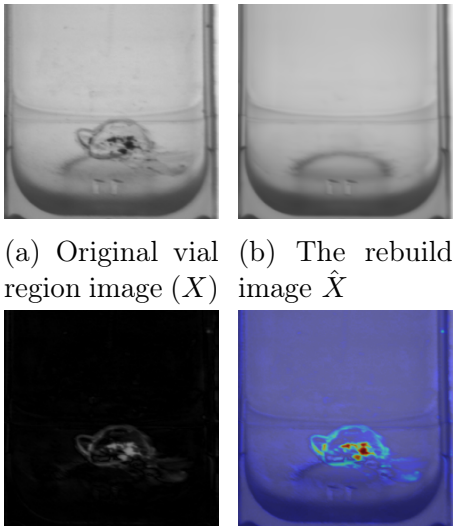


(c) The difference image



(d) The heatmap H superimposed

Figure E.7: Black spot on upper part of the vial's body.



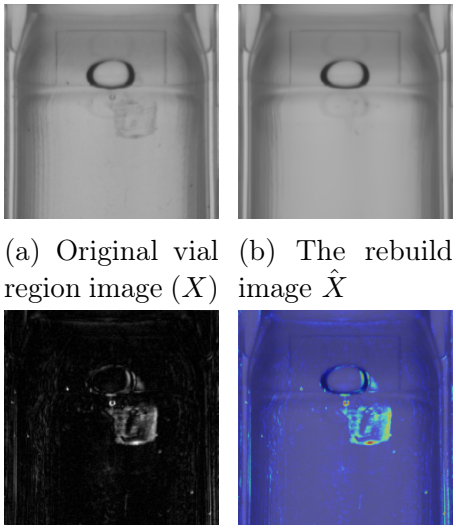
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.8: Burn in the lower part of the vial.



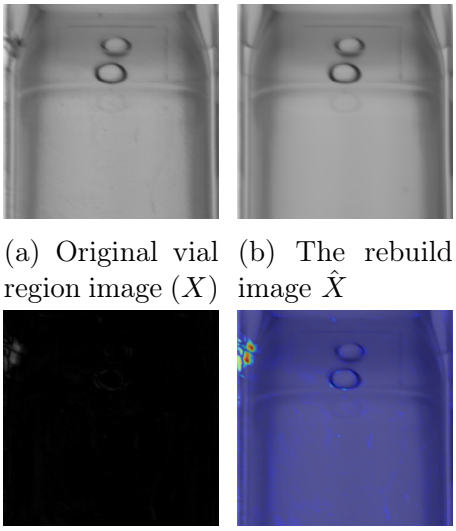
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.9: Light scratch near the product neck.



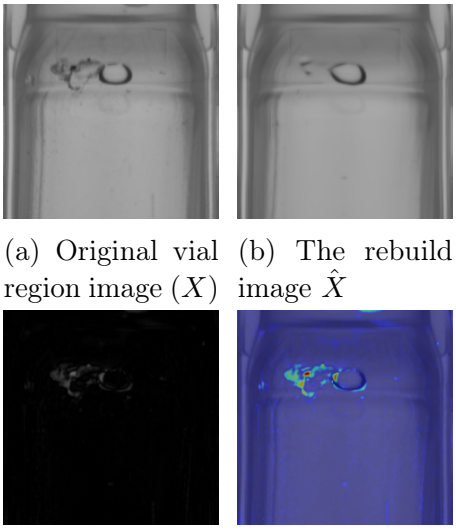
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.10: External imperfection caused by laser cutting of the product.



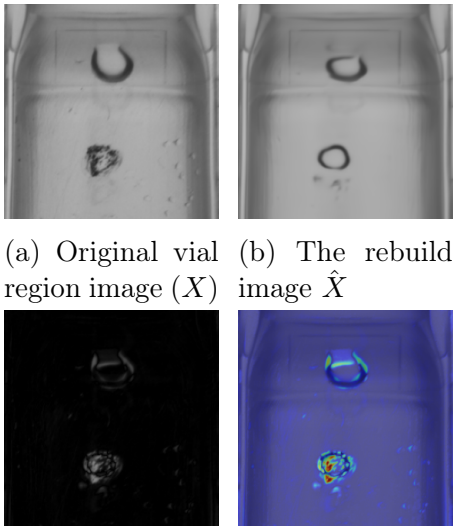
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.11: Deep scratch on the upper part of the vial's body.



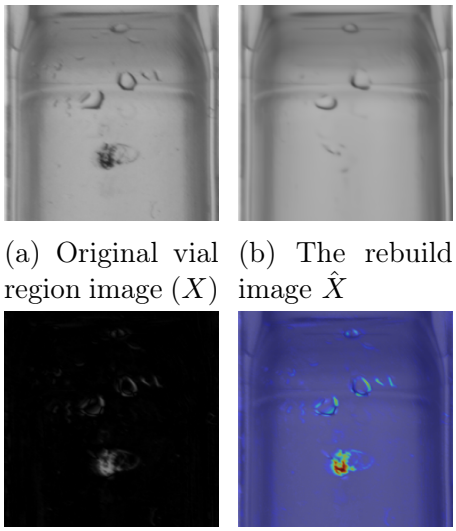
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.12: Burn on the vial's body. It is reproduced as a bubble, since the network does not know how to represent the defect features.



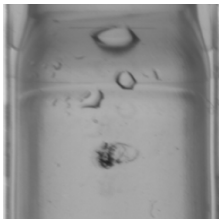
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

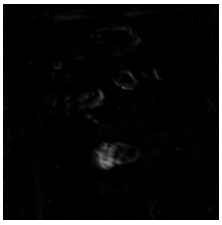
(d) The heatmap H superimposed

Figure E.13: Black spot on the vial's body

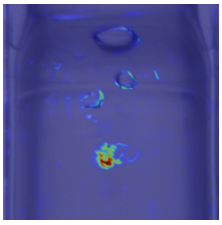


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

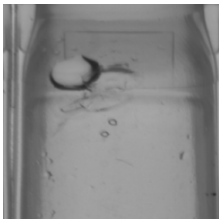


(c) The difference image



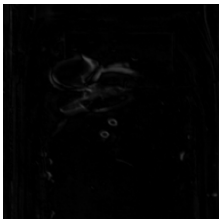
(d) The heatmap H superimposed

Figure E.14: Black spot of the vial's body

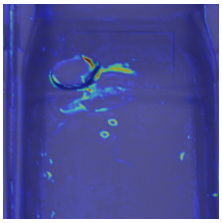


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

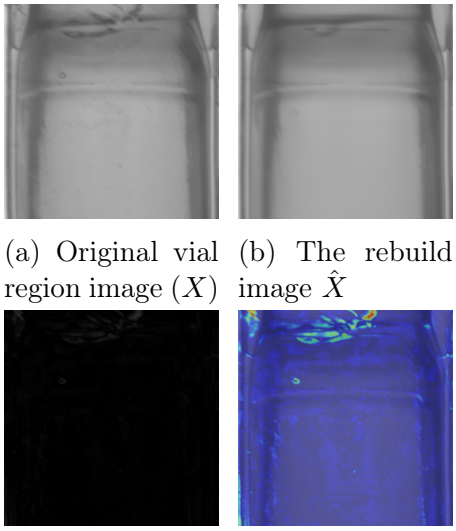


(c) The difference image



(d) The heatmap H superimposed

Figure E.15: Bump on the lower part of the vial's neck, near to a big bubble stuck in the internal layer of the BFS surface.



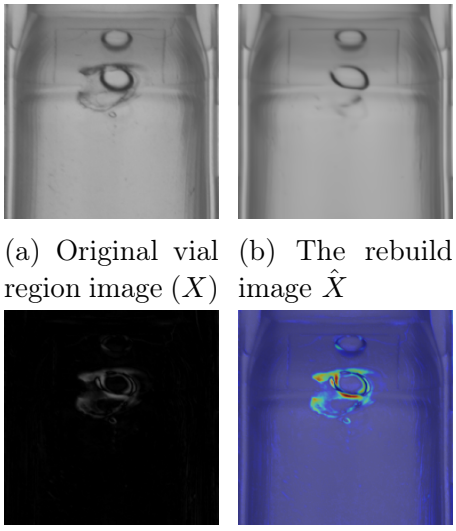
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.16: Light bump on the vial's neck



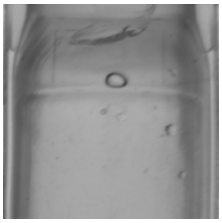
(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

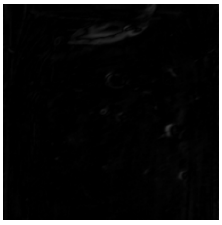
(d) The heatmap H superimposed

Figure E.17: Heavy deformation near the neck region, overlapped to a bubble that the network is still able to reproduce.

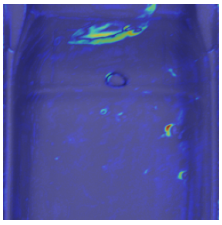


(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

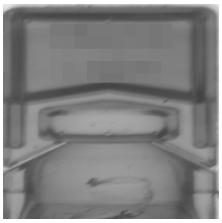


(c) The difference image

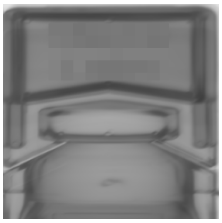


(d) The heatmap H superimposed

Figure E.18: Scratch on the vial's neck.

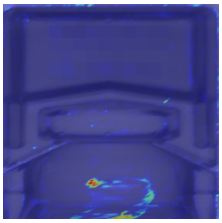


(a) Original vial region image (X)



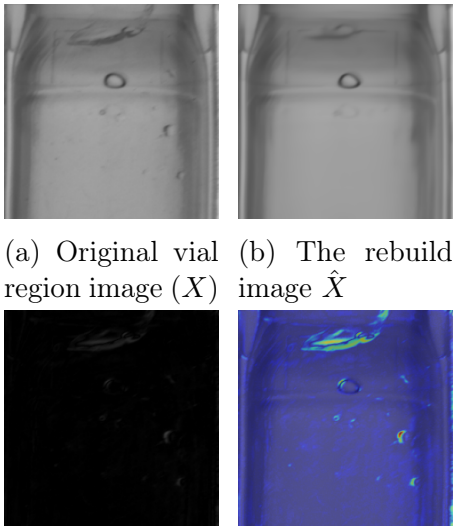
(b) The rebuild image $\hat{X}$

(c) The difference image



(d) The heatmap H superimposed

Figure E.19: Light bump on the lower part of the vial's cap region.



(a) Original vial region image (X)

(b) The rebuild image $\hat{X}$

(c) The difference image

(d) The heatmap H superimposed

Figure E.20: Scratch on the vial's neck.

Bibliography

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, 2017.

[2] Samet Akçay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. *ACM Asia Conference on Computer and Communications Security*, pages 622–633, 2018.

[3] Samet Akçay, Amir Atapour-Abarghouei, and Toby P Breckon. Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. In *International Joint Conference on Neural Networks (IJCNN)*, 2019.

[4] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Wasserstein gan. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 214–223. JMLR.org, 2017.

[5] Jane Austin and John Smith. Anomaly detection with conditioned denoising diffusion models. *Journal of Machine Learning Innovations*, 2021.

[6] Jaehyeok Bae, Jae-Han Lee, and Seyun Kim. Pni : Industrial anomaly detection using position and neighborhood information, 2023.

[7] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020.

[8] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The datacenter as a computer: Designing warehouse-scale machines*. Synthesis Lectures on Computer Architecture, 2018.

[9] Kilian Batzner, Lars Heckler, and Rebecca König. Efficientad: Accurate visual anomaly detection at millisecond-level latencies. 2024.

[10] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. *Deep Autoencoding Models for Unsupervised Anomaly Segmentation in Brain MR Images*, page 161–169. Springer International Publishing, 2019.

[11] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[12] Liron Bergman, Niv Cohen, and Yedid Hoshen. Deep nearest neighbor anomaly detection. *arXiv preprint arXiv:2002.10445*, 2020.

[13] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *International Journal of Computer Vision*, 130(4):947–969, Apr 2022.

[14] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019.

[15] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. *CoRR*, abs/1911.02357, 2019.

[16] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *arXiv preprint arXiv:1807.02011*, 2018.

[17] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and Jared D Kaplan. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[18] Sai Bulusu, Bhavya Kailkhura, Bo Li, Kush R Varshney, and Dawn Song. Anomalous example detection in deep learning: A survey. *IEEE Access*, 8:132330–132347, 2020.

[19] Jonathan Burez and Dirk Van den Poel. Handling class imbalance in customer churn prediction. *Expert systems with applications*, 36(3):4626–4636, 2009.

[20] Dongyue Chen, Lingyi Yue, Xingya Chang, Ming Xu, and Tong Jia. Nm-gan: Noise-modulated generative adversarial network for video anomaly detection. *Pattern Recognition*, 116:107969, 2021.

[21] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[22] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[23] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[24] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[25] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *CoRR*, abs/2005.02357, 2020.

[26] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020.

[27] OpenCV Contributors. Blob detection in opencv. OpenCV Documentation, 2023. Available: `https://docs.opencv.org/master/d0/d7a/classcv_1_1SimpleBlobDetector.html`.

[28] Mark Corbett and Aaron Kennedy. Quality control in the pharmaceutical industry: A computer vision perspective. *Pharmaceutical Engineering*, 42:34–45, 2022.

[29] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[30] Sanjoy Dasgupta and Daniel Hsu. On-line estimation with the multivariate gaussian distribution. In Nader H. Bshouty and Claudio Gentile, editors, *Learning Theory*, pages 278–292, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[31] Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Marc'Aurelio Ranzato Andrew Senior Mao, Paul Tucker, Ke Yang, and Andrew Y Ng. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[32] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*, 2021.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[34] Zhiquan Ding, Yuejin Zhang, Chenxin Zhu, Guolong Zhang, Xiong Li, Nan Jiang, Yue Que, Yuanyuan Peng, and Xiaohui Guan. Cat-unet: An enhanced u-net architecture with coordinate attention and skip-neighborhood attention transformer for medical image segmentation. *Information Sciences*, 670:120578, 2024.

[35] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[36] Jeff Donahue, Philipp Krahenbuhl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2017.

[37] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2021.

[38] James Dura and Kiran Shah. Challenges in deep learning-based defect detection in pharmaceutical manufacturing. *Journal of Pharmaceutical Sciences*, 108(5):1596–1605, 2019.

[39] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[40] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[41] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021.

[42] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[43] Niccolò Ferrari, Michele Fraccaroli, and Evelina Lamma. Grd-net: Generative-reconstructive-discriminative anomaly detection with region of interest attention module. *International Journal of Intelligent Systems*, 2023.

[44] Niccolò Ferrari, Nicola Zanarini, Michele Fraccaroli, Alice Bizzarri, and Evelina Lamma. Integration of deep generative anomaly detection algorithm in high-speed industrial line. *Submitted to Engineering Applications of Artificial Intelligence (preprint)*, 2024.

[45] Michele Fraccaroli, Alice Bizzarri, Paolo Casellati, and Evelina Lamma. Exploiting cnn's visual explanations to drive anomaly detection. Submitted to Applied Intelligence, Springer.

[46] Michele Fraccaroli, Alice Bizzarri, Paolo Casellati, and Evelina Lamma. Cross entropy overlap distance. Accepted and Presented at ITAL-IA 2022, workshop on AI for Industry, feb 2022.

[47] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3146–3154, 2019.

[48] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980.

[49] Florian Gälà, Fabien Baradel, Alessandro Bambi, and Vittorio Ferrari. A survey on visual anomaly detection. *Pattern Recognition*, 114:107849, 2021.

[50] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1180–1189, 2015.

[51] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[52] Kaan Gokcesu and Hakan Gokcesu. Generalized huber loss for robust learning and its efficient minimization for a robust statistics, 2021.

[53] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. *CoRR*, abs/1805.10917, 2018.

[54] Dong Gong, Ling Liu, Vuong Le, Budhaditya Saha, Mohamed Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality

to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.

[55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[56] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. 2014.

[57] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. *WACV*, 2022.

[58] Denis Gudovskiy and Luca Rigazio. Cflow: Conditional generative flow models for images and 3d point cloud anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1910–1919, 2022.

[59] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22):2402–2410, 2016.

[60] Jia Guo, Shuai Lu, Lize Jia, Weihang Zhang, and Huiqi Li. Recontrast: Domain-specific anomaly detection via contrastive reconstruction. 2023.

[61] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[62] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021.

[63] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9729–9738, 2020.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *European conference on computer vision*, pages 630–645, 2016.

[66] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joe Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings, 2022.

[67] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017*, 2016.

[68] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[69] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations*, 2017.

[70] John L Hennessy and David A Patterson. A new golden age for computer architecture. *Communications of the ACM*, 62(2):48–60, 2019.

[71] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[72] Jong Chul (JC) Ho, A. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6840–6851. NeurIPS, 2020.

[73] Jong Chul (JC) Ho and Tim Salimans. Class-conditional denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6740–6751. NeurIPS, 2021.

[74] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[75] Seunghoon Hong, Jinwoo Oh, Honglak Lee, and Bohyung Han. Deep learning for domain adaptation: An overview. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[76] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[77] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[78] Wenqian Jiang, Yang Hong, Xin He, and Cheng Cheng. A gan-based anomaly detection approach for imbalanced industrial time series. *IEEE Access*, PP:1–1, 09 2019.

[79] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12, 2017.

[80] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2018.

[81] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[82] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2019.

[83] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, 2018.

[84] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[85] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

[86] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

[87] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012.

[89] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[90] Viet-Tuan Le and Yong-Guk Kim. Attention-based residual autoencoder for video anomaly detection. *Applied Intelligence*, 53(3):3240–3254, Feb 2023.

[91] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[92] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[93] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks, 2018.

[94] Dan Li, Dacheng Chen, Lei Shi, Baihong Jin, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, 2019.

[95] Shiyu Liang, Yixuan Li, and R. Srikant. Principled detection of out-of-distribution examples in neural networks. volume abs/1706.02690, 2017.

[96] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[97] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection. *CoRR*, abs/2010.03759, 2020.

[98] XYZ Liu and et al. Glass: Gradient layered anomaly synthesis strategy with semantic features. *arXiv preprint arXiv:2111.12345*, 2021.

[99] Yanning Liu, Aditya Jain, Clara Eng, David H. Way, Kang Lee, Peggy Bui, Ken Kanada, and Gabriel de la Torre. A deep learning system for differential diagnosis of skin diseases. *Nature Medicine*, 25(6):900–908, 2019.

[100] Zhikang Liu, Yiming Zhou, Yuansheng Xu, and Zilei Wang. Simplenet: A simple network for image anomaly detection and localization. 2023.

[101] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[102] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[103] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations*, 2016.

[104] Min Lu, Bin Zhou, and Zhiyong Bu. Attention-empowered residual autoencoder for end-to-end communication systems. *IEEE Communications Letters*, 27(4):1140–1144, 2023.

[105] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. *Proc. ICML*, 30(1), 2013.

[106] S.B. Maind and P Wankar. Research paper on basic of artificial neural network. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2:96–100, 01 2014.

[107] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[108] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection, 2021.

[109] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[110] Umberto Michelucci. An introduction to autoencoders. *CoRR*, abs/2201.03898, 2022.

[111] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[112] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. IEEE, June 2021.

[113] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[114] David Mutz, Johannes Krause, Matthias Preuß, Thomas Grohmann, and Steffen Schuler. Vision-based inspection technologies in pharmaceutical manufacturing: A review. *Journal of Pharmaceutical Innovation*, 17:658–675, 2022.

[115] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[116] Paolo Napoletano, Flavio Piccoli, and Raimondo Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors*, 18(1):209, 2018.

[117] Allen Newell. Perceptrons. an introduction to computational geometry. marvin minsky and seymour papert. m.i.t. press, cambridge, mass., 1969. vi + 258 pp., illus. cloth, 12; *paper*,4.95. *Science*, 165(3895):780–782, 1969.

[118] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8520–8531. ICML, 2021.

[119] Zijian Niu, Ke Yu, and Xiaofei Wu. Lstm-based vae-gan for time-series anomaly detection. *Sensors*, 20:3738, 07 2020.

[120] World Health Organization. Quality assurance of pharmaceuticals: a compendium of guidelines and related materials. 2024.

[121] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[122] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. 2019.

[123] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2898–2906, 2019.

[124] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[125] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. *Advances in neural information processing systems*, 31, 2018.

[126] Chinmay Prabhakar, Hongwei Bran Li, Jiancheng Yang, Suprosana Shit, Benedikt Wiestler, and Bjoern Menze. Vit-ae++: Improving vision transformer autoencoder for self-supervised medical image representations. 2023.

[127] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil Lawrence. A dataset shift in machine learning. *Journal of Machine Learning Research*, 6:45–50, 2009.

[128] Panagiotis Radoglou Grammatikis, Panagiotis Sarigiannidis, Georgios Efstathopoulos, and Emmanouil Panaousis. Aries: A novel multivariate intrusion detection system for smart grid. *Sensors*, 20, 09 2020.

[129] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[130] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, and A. Radford. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. ICML, 2021.

[131] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[132] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection, 2019.

[133] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, pages 91–99, 2015.

[134] Oliver Rippel, Patrick Mertens, and Dorit Merhof. Modeling the distribution of normal data in pre-trained deep features for anomaly detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6726–6733. IEEE, 2021.

[135] H Risken. *Fokker-Planck Equation*. Springer, Berlin, Heidelberg, 2nd edition edition, 1996.

[136] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015.

[137] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9424–9433, 2021.

[138] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1907–1916, 2021.

[139] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[140] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3379–3388, 2018.

[141] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainability in deep learning: A survey. *arXiv preprint arXiv:1708.08296*, 2017.

[142] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *IPMI*, pages 146–157, 2017.

[143] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[144] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[145] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, 2018.

[146] Walter A. Shewhart. *Economic Control of Quality of Manufactured Product*. Van Nostrand Company, Inc., New York, 1931.

[147] Nina Shvetsova, Bart Bakker, Irina Fedulova, Heinrich Schulz, and Dmitry V. Dylov. Anomaly detection in medical imaging with deep perceptual autoencoders. *IEEE Access*, 9:118571–118583, 2021.

[148] Ilias Siniosoglou, Panagiotis Radoglou Grammatikis, George Efstathopoulos, Panagiotis Fouliras, and Panagiotis Sarigiannidis. A unified deep learning anomaly detection and classification approach for smart grid environments. *IEEE Transactions on Network and Service Management*, PP, 05 2021.

[149] Jane Smith and Others. Enhancing explainability in machine learning models for pharmaceutical applications. *Journal of Pharmaceutical Innovation*, 39:1012–1029, 2024.

[150] John et al. Smith. Advanced machine learning techniques for industrial applications. *Journal of Industrial Technology*, 29:154–168, 2024.

[151] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. 2015.

[152] Ludvig (L.) Song and Stefano Ermon. Improved techniques for training score-based generative models. In *International Conference on Learning Representations (ICLR)*. ICLR, 2020.

[153] Peter Sperl and et al. Efficient anomaly detection with generative adversarial networks. *arXiv preprint arXiv:2011.11123*, 2020.

[154] Xiao Sun, Peng Zhao, and Xiaoyang Liu. Deep learning-based vision inspection systems for the pharmaceutical industry. *Journal of Biomedical Optics*, 25(6):061607, 2020.

[155] Halcon Development Team. Blob analysis in halcon. Halcon Software Documentation, 2023. Available: `https://www.mvtec.com/products/halcon`.

[156] Yonglong Tian, Olivier J. Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data, 2021.

[157] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018.

[158] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[159] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[160] Shashanka Venkataramanan, Kuan-Chuan Peng, Rajat Vikram Singh, and Abhijit Mahalanobis. Attention guided anomaly localization in images. In *European Conference on Computer Vision*, pages 485–503. Springer, 2020.

[161] Nithya Venkatesh, Ayan Chakraborty, and Ratheesh Govindarajan. Computer vision for pharmaceutical manufacturing: A comprehensive review. *International Journal of Pharmaceutical Investigation*, 10(2):43–58, 2020.

[162] Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H. Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2017.

[163] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[164] Chathurika Wickramasinghe, Daniel Marino, and Milos Manic. Resnet autoencoders for unsupervised feature learning from high-dimensional data: Deep models resistant to performance degradation. *IEEE Access*, PP:1–1, 03 2021.

[165] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[166] Xuan Xia, Xizhou Pan, Nan Li, Xing He, Lin Ma, Xiaoguang Zhang, and Ning Ding. Gan-based anomaly detection: A review. *Neurocomputing*, 493:497–535, 2022.

[167] Xin Xie, Yuhui Huang, Weiye Ning, Dengquan Wu, Zixi Li, and Hao Yang. Rdad: A reconstructive and discriminative anomaly detection model based on transformer. *International Journal of Intelligent Systems*, 37(11):8928–8946, 2022.

[168] Jihoon Yu, Kwanghoon Sohn, Wonjun Kim, and Seunghoon Yoon. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows. *CVPR*, 2021.

[169] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[170] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Draem – a discriminatively trained reconstruction embedding for surface anomaly detection, 2021.

[171] Riccardo Zese, Elena Bellodi, Michele Fraccaroli, Fabrizio Riguzzi, and Evelina Lamma. *Neural Networks and Deep Learning Fundamentals*, pages 23–42. Springer International Publishing, 2022.

[172] Zerui Zhang, Zhichao Sun, Zelong Liu, Bo Du, Rui Yu, Zhou Zhao, and Yongchao Xu. Spatial-aware attention generative adversarial network for semi-supervised anomaly detection in medical image, 2024.

[173] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[174] Guangxuan Zhu, Hongbo Zhao, Haoqiang Liu, and Hua Sun. A novel lstm-gan algorithm for time series anomaly detection. pages 1–6, 10 2019.

[175] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[176] Xiaojin Zhu and Andrew B Goldberg. *Introduction to semi-supervised learning*. Morgan & Claypool, 2009.

[177] Simone Zini, Simone Bianco, and Raimondo Schettini. Deep residual autoencoder for quality independent JPEG restoration. *CoRR*, abs/1903.06117, 2019.

[178] Simone Zini, Simone Bianco, and Raimondo Schettini. Deep residual autoencoder for blind universal jpeg restoration. *IEEE Access*, 8:63283–63294, 2020.

Author's Publications

Publications

All works developed during the PhD and described in this thesis have led to the publications listed below.

- International Journals:
  - Niccolò Ferrari, Michele Fraccaroli, Evelina Lamma GRD-Net: Generative-Reconstructive-Discriminative Anomaly Detection with Region of Interest Attention Module, International Journal of Intelligent Systems, Wiley, 2023. [43]
- Submitted:
  - Ferrari Niccolò, Zanarini Nicola, Fraccaroli Michele, Bizzarri Alice, Lamma Evelina, Integration of Deep Generative Anomaly Detection Algorithm in High-Speed Industrial Line. Available at SSRN: https://ssrn.com/abstract=4858664 or http://dx.doi.org/10.2139/ssrn.4858664. Submitted to Engineering Applications of Artificial Intelligence (preprint). [44]